RAPID AND ACCURATE FAULT DETECTION FOR UNCERTAIN NONLINEAR SYSTEMS USING ADVANCED SET-BASED STATE ESTIMATION TECHNIQUES

A Dissertation
Presented to
The Academic Faculty

By

Xuejiao Yang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Chemical & Biomolecular Engineering

Georgia Institute of Technology

December 2020

RAPID AND ACCURATE FAULT DETECTION FOR UNCERTAIN NONLINEAR SYSTEMS USING ADVANCED SET-BASED STATE ESTIMATION TECHNIQUES

Approved by:

Dr. Joseph K. Scott, Advisor
School of Chemical and Biomolecular Engineering
Georgia Institute of Technology

Dr. Martha A. Grover
School of Chemical and Biomolecular Engineering
Georgia Institute of Technology

Dr. Mark P. Styczynski
School of Chemical and Biomolecular Engineering
Georgia Institute of Technology

Dr. Matthew J. Realff
School of Chemical and Biomolecular Engineering
Georgia Institute of Technology

Dr. Samuel Coogan
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: September 15, 2020

ACKNOWLEDGEMENTS

I would like to thank the School of Chemical and Biomolecular Engineering at Georgia Institute of Technology for offering me a valuable opportunity to complete my study here. I will never forget to thank Dr. Joseph K. Scott, my advisor, for whom without his patience, guidance, knowledge, and understanding, I would not have the achievements during the last five years, especially my dissertation. I also remain grateful for all my group members, who have helped me with my life and research in the past years. You are all special to me. Finally, and most importantly, huge thanks to my family, my parents, and my friends for all their supports.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In applications such as wind energy, industrial robotics, and chemical processing, increases in complexity and automation have made component malfunctions and other abnormal events (i.e., faults) an ever-present threat to safety and reliability. Thus, fault detection algorithms have become an essential feature of modern control systems, leading to significant decreases in downtime, maintenance costs, and catastrophic failures. However, while well-established statistical methods are effective in many cases, they often fail to make the critical distinction between faults and normal process disturbances. An attractive alternative is to exploit detailed process models that, at least in principle, can be used to characterize the outputs consistent with normal operation, providing a rigorous basis for fault detection. Methods that furnish a guaranteed enclosure of these outputs (e.g., using set-based state estimators) are particularly attractive because they eliminate the possibility of costly false alarms and provide better trade-offs between false alarms and missed faults. However, such methods are currently impractical for systems with strong nonlinearities or large uncertainties. For such systems, existing set-based estimation techniques often produce enclosures that are far too conservative to be useful for fault detection, or avoid this only at excessive computational cost. Thus, there is a critical need for advanced algorithms that can rapidly detect faults for realistic nonlinear systems, and do so rigorously in the presence of disturbances, measurement noise, and large model uncertainties.

In this thesis, we develop an advanced set-based state estimation method for uncertain nonlinear systems, and demonstrate its application to provide fast and accurate fault detection for such systems. Our proposed estimation method is performed recursively in two steps. First, the prediction step computes an enclosure of the possible model outputs under uncertainty over one discrete time step. Next, the correction step uses the process measurements to update this enclosure by eliminating regions that are not consistent with the measurements. In contrast to existing set-based estimation methods, our prediction step makes use of our previously developed continuous-time differential inequalities (DI) method and extends it to discrete-time systems. The DI method uses very efficient interval computations, but is effective at mitigating some key sources of conservatism typically associated with such computations in discrete-time systems by exploiting redundant model equations, which can be easily found in many representative reaction and separation models. Moreover, we make use of past process measurements in a novel way in the prediction step, potentially leading to further improvements in bound accuracy. Our results demonstrate that, for a variety of systems of practical interest, the proposed prediction step in the state estimation algorithm leads to dramatically tighter enclosures of the states, with only modest additional computational cost relative to standard interval methods. Moreover, by combining the proposed correction step with the prediction method, this guaranteed state estimation algorithm largely increases the accuracy of the estimated state sets and is suitable for online applications. The numerical results show that this method produces state estimates with significantly higher accuracy and efficiency than state-of-the-art zonotopic methods for a challenging nonlinear chemical reactor model. Finally, we apply the resulting estimators to achieve significantly faster and more accurate fault detection than is achievable with existing fault detection methods. The proposed approach is demonstrated to achieve high accuracy and eliminate false alarms using a range of examples with comparisons to existing state-of-the-art data-based and model-based fault detection algorithms.

CHAPTER 1

INTRODUCTION AND BACKGROUND

1.1 Overview

Due to the level of complexity, integration, and automation in modern chemical processes, robotics systems, and power systems, faults such as equipment malfunctions and failures pose a serious threat to safe and profitable operation. In the United States alone, the lack of high-performance fault detection systems results in 20 billion dollars in losses annually in the chemical industry [1]. Classical fault detection (FD) methods exploit historical data and are well established for various systems. However, these data-based methods cannot rigorously distinguish faults from system disturbances. Therefore, a lack of high-quality historical data can result in alarms in normal situations (i.e., false alarms) or failures to detect faults (i.e., missed faults). A promising alternative is to use a first-principles model to detect faults by comparing the model predictions to the outputs observed from the real process. Set-based FD is a particularly useful model-based approach where a fault is declared whenever the measured output lies outside of a rigorous enclosure of all possible model outputs subject to disturbances and other model uncertainties. Such an enclosure can be computed using a set-based state estimation method. This approach eliminates false alarms while still having high sensitivity to faults, provided that the set-based state estimator is accurate. However, modern set-based state estimation methods are either too computationally demanding for online FD or produce enclosures that are too conservative for effective FD.

To address these challenges, this thesis develops advanced set-based state estimation methods for nonlinear discrete-time systems with large uncertainties and applies them for online set-based fault detection. Set-based estimation is commonly performed recursively in two steps. Given an enclosure of the possible system states at the current time, the prediction step uses the model to compute an enclosure of all possible states in the next discrete time step under uncertainty. Next, the correction step updates this enclosure by eliminating regions that are inconsistent with the process measurements. Prediction requires propagating sets through nonlinear uncertain dynamics, which is a major source of conservatism in existing methods. The correction step requires bounding the intersection of two complex sets, which is also nontrivial. This thesis develops a set-based FD method by addressing these challenges in set-based estimation first. Our overall contributions are achieved through the following specific objectives:

1. Develop an accurate and efficient reachability analysis method for the prediction step in set-based state estimation
2. Develop a fast and accurate set-based state estimation algorithm by combining the prediction step with an effective measurement correction step
3. Develop an effective online set-based FD algorithm using the advanced set-based state estimation method. The proposed algorithm guarantees no false alarms and has significantly higher fault sensitivity than existing set-based FD methods.

The resulting algorithm enables faults to be detected in highly nonlinear and uncertain systems with significantly higher speed and accuracy than is currently possible. Therefore, this work could help reduce process downtime, financial losses, and safety risks caused by equipment malfunctions and other abnormal operations in a range of applications, such as pharmaceutical processes, autonomous vehicles, wind turbines, etc. Objective 1–3 are introduced in detail in Sections 1.2–1.4 below.

In addition to these main objectives, this thesis also has two separate contributions. Firstly, Chapter 5 provides a detailed review and comparison of existing zonotope order reduction methods. Zonotopes are a class of centrally symmetric convex polytopes that have a variety of computational advantages and are widely used in algorithms for state estimation, fault detection, and elsewhere. However, many operations on zonotopes yield results with higher complexity than their arguments [2], which is a serious limitation, particularly for recursive algorithms. Order reduction methods bound a given zonotope within another of lower complexity, and are essential for many algorithms using zonotopes. The results of our comparison provide valuable guidance for designing set-based estimation and control algorithms that more effectively balance accuracy with computational cost.

Secondly, Chapter 6 proposes a method for using reachability analysis to verify the safety of autonomous vehicles. This method is similar to the proposed reachability analysis method in Objective 1, but it applies to continuous-time systems. This algorithm enables efficient computation of accurate bounds on the possible vehicle trajectories under uncertainty, which can potentially be applied for online collision avoidance.

These two additional contributions are discussed in more detail in Sections 1.5 and 1.6 below.

1.2 Reachability Analysis

This section introduces Objective 1 of this thesis in more detail. Consider the following discrete-time system, where $\mathbf{x}_k$ is the state, $\mathbf{w}_k$ is the disturbance, $\mathbf{c}_0$ is the initial condition, and the time horizon is $\mathbb{K} \equiv \{0, \ldots, K\}$:

$$\mathbf{x}_{k+1} = \mathbf{h}\left(k, \mathbf{x}_k, \mathbf{w}_k\right), \quad \mathbf{x}_0 = \mathbf{c}_0. \tag{1.1}$$

Let C_0 and W be given compact sets of admissible initial conditions and disturbances, respectively, so that $\mathbf{c}_0 \in C_0$ and $\mathbf{w}_k \in W$, $\forall k \in \mathbb{K}$. Furthermore, define the sequence shorthand $\mathbf{w}_{0:K} = (\mathbf{w}_0, \ldots, \mathbf{w}_K)$, $W_{0:K} \equiv W \times \cdots \times W$, $\mathbf{x}_{0:K} = (\mathbf{x}_0, \ldots, \mathbf{x}_K)$, and

$\mathbb{R}^{n_x}_{0:K} \equiv \mathbb{R}^{n_x} \times \cdots \times \mathbb{R}^{n_x}$. We call $(\mathbf{c}_0, \mathbf{w}_{0:K}, \mathbf{x}_{0:K}) \in C_0 \times W_{0:K} \times \mathbb{R}^{n_x}_{0:K}$ a solution of (1.1) if it satisfies (1.1) for all $k \in \{0, \dots, K-1\}$. The *reachable set* of the discrete-time system (1.1) is defined for every $k \in \mathbb{K}$ by

$$\begin{aligned}\mathcal{R}_k \equiv &\{\mathbf{z} \in \mathbb{R}^{n_x} : \exists \text{ a solution } (\mathbf{c}_0, \mathbf{w}_{0:K}, \mathbf{x}_{0:K}) \in C_0 \times W_{0:K} \\ &\times \mathbb{R}^{n_x}_{0:K} \text{ of (1.1) satisfying } \mathbf{x}_k = \mathbf{z}\}.\end{aligned}$$

In words, the reachable set at time k is the set of all states of (1.1) at k that can be obtained with an admissible initial condition in C_0 and sequence of disturbance vectors in $W_{0:k-1}$. Objective 1 of this thesis is to develop a method for efficiently computing an accurate enclosure of the reachable set for all $k \in \mathbb{K}$.

1.2.1 Motivation and Existing Methods

Enclosing reachable sets is a critical step in set-based state estimation [3, 4], which is in turn used in a variety of robust control and fault detection algorithms [5, 6, 7, 8]. Reachable sets are also widely used in safety verification, motion planning, design space construction, and many other applications [9, 10, 11].

A number of effective algorithms are available for bounding the reachable sets of discrete-time linear systems [12, 13, 14, 2]. However, for nonlinear systems, computing accurate enclosures remains a significant challenge, especially when enclosures must be computed rapidly online. In essence, propagating a reachable set enclosure X_k from time k to $k+1$ is equivalent to bounding the image of X_k under the nonlinear vector function defining the dynamics. This can be done efficiently using interval arithmetic [15], but the resulting enclosure is often very weak. A tighter interval enclosure can be obtained by partitioning X_k, but this is much more costly [15, 16]. For polynomial systems, tighter bounds on the image of X_k have been obtained using optimization formulations such as linear, semidefinite, and DC programming [17, 18, 19]. In [20], a new approach for efficiently optimizing polynomials over parallelotopes using the Bernstein basis is applied, and in [11, 21], polynomial dynamics are represented as linear fractional transformations and bounded using the skewed structured singular value. However, optimization-based approaches are generally not suitable for online applications. A faster approach is to propagate X_k from k to $k+1$ by first considering a local linearization of the dynamics and then adding a rigorous bound on the linearization error [22, 23]. This strategy exploits efficient set-based calculations that are possible in the linear case (e.g., using zonotopes), but can suffer from large linearization errors for nonlinear systems.

1.2.2 Contribution

In this work, we develop a new class of methods for discrete-time reachability analysis that is motivated by the theory of differential inequalities (DI). This theory pertains to continuous-time systems, rather than the discrete-time systems of interest here, and is the basis for some very effective reachability analysis methods in the continuous-time setting (continuous-time reachability methods are reviewed more comprehensively in Section 1.6). Given a continuous-time systems, the basic DI method uses interval arithmetic to construct bounding differential equations that furnish time-varying interval bounds as their solutions [24]. Like discrete-time interval methods (without partitioning), this produces bounds at low cost, but can be very conservative. However, several advanced DI methods have recently been developed that largely retain the efficiency of the original method while providing much tighter enclosures (see §2.1) [25, 24, 26, 27]. Thus, there is significant motivation to extend these approaches to discrete-time.

However, DI-based reachability methods are based on theoretical arguments that are only valid for continuous-time systems. Specifically, DI theory depends critically on the fact that, for continuous-time systems, a trajectory cannot leave a set X without crossing its boundary. Thus, to propagate a reachable set enclosure forward in time, it suffices to consider the behavior of the vector field on its boundary [28]. Unfortunately, this is not true in discrete-time, and this precludes any straightforward analogue of the DI approach for general discrete-time systems.

However, in practice, most discrete-time systems of interest are derived as approximations of an underlying continuous-time system. If such an approximation is accurate enough, it is sensible to expect that a discrete-time DI method might produce valid bounds. Following this idea, Chapter 2 develops a novel discrete-time extension of the basic DI bounding algorithm and proves that it produces valid reachable set enclosures provided that the discrete-time dynamics satisfy a certain monotonicity condition. As an important special case, we then show that any system derived by forward Euler discretization of a continuous-time model will satisfy this monotonicity requirement whenever the discretization step size is below an upper bound. This step size bound can be easily computed in advance, and is no more restrictive than the step size required to preserve basic physical properties of the solution, such as non-negativity [29]. Next, the advanced DI methods in [24, 26] are also extended to discrete-time systems and proven to be valid under a tighter step size restriction. Numerical comparisons show that that these discrete-time DI algorithms offer significant advantages over the standard discrete-time interval method and two popular methods using zonotopes [22, 23] in terms of both speed and accuracy.

1.3 Set-Based State Estimation

This section introduces Objective 2 of this thesis in more detail. Consider system (1.1) again, but now suppose that at each time k we obtain a measured output $\mathbf{y}_k$ with measurement error $\mathbf{v}_k$. This leads to the following discrete-time system:

$$\mathbf{x}_{k+1} = \mathbf{h}\left(k, \mathbf{x}_k, \mathbf{w}_k\right), \tag{1.2a}$$

$$\mathbf{x}_0 = \mathbf{c}_0, \tag{1.2b}$$

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k, \mathbf{v}_k). \tag{1.2c}$$

As with $\mathbf{c}_0$ and $\mathbf{w}_k$, we assume that $\mathbf{v}_k$ is time-varying and unknown but bounded within a given compact set V, so that $\mathbf{v}_k \in V$, $\forall k \in \mathbb{K}$.

The objective is to develop a fast and accurate set-based state estimation algorithm for (1.2). In contrast to conventional state estimation, which aims to compute a single best estimate for the current state, set-based state estimation aims to compute a set that rigorously encloses all states consistent with the given model and the observed outputs up to the present time.

1.3.1 Motivations and Existing Methods

Set-based state estimation is an essential step in many algorithms for robust control [30, 31], fault detection and diagnosis [8, 32, 7], fault tolerant control [33], safety verification and collision avoidance [34], and others. In this thesis, we are primarily interested in the application of set-based estimation to fault detection. There, set-based state estimation is used to compute a rigorous enclosure of the set of all outputs at time k that are consistent with the fault-free process model, the bounded system uncertainties, and all past measurements. Faults are then detected by testing if the measured output lies within this set. This provides a rigorous way to distinguish the effects of faults from those of admissible disturbances, measurement noises, and other uncertainties. However, this application requires a highly accurate set-based state estimator because overly conservative enclosures will lead to very low fault sensitivities. Therefore, there is a critical need to study effective set-based state estimation approaches.

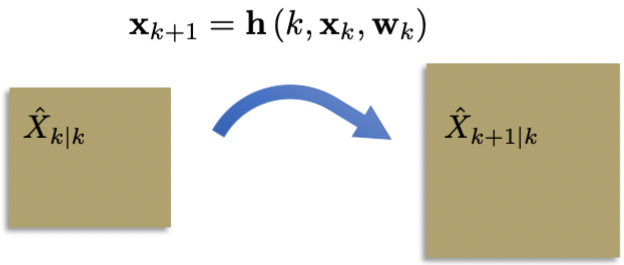
$\mathbf{x}_{k+1} = \mathbf{h}(k, \mathbf{x}_k, \mathbf{w}_k)$
$\hat{X}_{k|k}$
$\hat{X}_{k+1|k}$

Figure 1.1: Prediction: Computing an interval enclosure $\hat{X}_{k+1|k}$ of states reachable at $k+1$ from $\hat{X}_{k|k}$ under given dynamics.

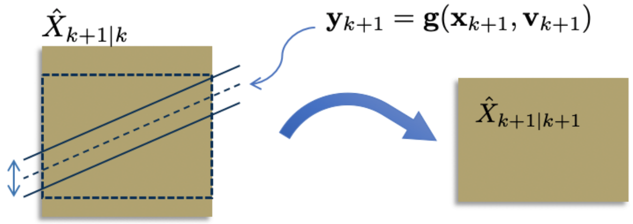
$\hat{X}_{k+1|k}$
$\mathbf{y}_{k+1} = \mathbf{g}(\mathbf{x}_{k+1}, \mathbf{v}_{k+1})$
$\hat{X}_{k+1|k+1}$

Figure 1.2: Correction: Computing an interval enclosure of the states in the prediction set $\hat{X}_{k+1|k}$ that are consistent with the measurement $\mathbf{y}_{k+1}$.

A variety of set-based state estimation algorithms have been proposed for nonlinear discrete-time systems. In most approaches, estimation is done recursively in two steps. Given a set of consistent states $X_{k|k}$ at time k, the *prediction* step aims to computes the set $X_{k+1|k}$ of states reachable at $k+1$ from $X_{k|k}$ under the given dynamics (1.2a). Next, in the *correction* step, $X_{k+1|k}$ is refined to produce the corrected set $X_{k+1|k+1}$ by eliminating regions of $X_{k+1|k}$ that are inconsistent with the observed output at $k+1$. Unfortunately, the exact sets $X_{k+1|k}$ and $X_{k+1|k+1}$ can be arbitrarily complex and are not computable. Thus, it is necessary to compute enclosures $\hat{X}_{k+1|k} \supset X_{k+1|k}$ and $\hat{X}_{k+1|k+1} \supset X_{k+1|k+1}$ in the form of simpler sets such as interval, ellipsoids, or convex polytopes. Figures 1.1 and 1.2 give an example of prediction and correction steps using interval enclosures. Figure 1.1 shows the prediction step, which computes an interval enclosure $\hat{X}_{k+1|k}$ of states reachable at $k+1$ from $\hat{X}_{k|k}$ under (1.2a). Figure 1.1 shows the correction step, which encloses states that are consistent with $\hat{X}_{k+1|k}$ and the measurement $\mathbf{y}_{k+1}$.

The prediction step requires bounding the image of $\hat{X}_{k|k}$ under the nonlinear vector function defining the dynamics. Therefore, this step is equivalent to a single time-step of the reachability analysis problem described in Section 1.2, and any of the methods described there can be applied. A variety of prediction steps have been proposed using enclosures with different geometries, including intervals, ellipsoids, parallelotopes, polytopes, zonotopes, and constrained zonotopes [15, 35, 13, 14, 2, 4]. However, as discussed in Section 1.2, computing accurate reachable set enclosures remains a significant challenge, especially when the algorithm has to be efficient enough for online systems. Thus, the prediction step remains a major cause of conservatism in existing set-based state estimation methods.

The correction step also suffers from large overestimation errors or high computational costs, particularly for nonlinear systems with large uncertainties. In [36, 22], the correction step requires bounding the intersection of an ellipsoid or a zonotope with a measurement set, which remains a source of significant overestimation using simple heuristics. Interval partitions are exploited in [15] but lead to exponential run-times. In [35, 23, 37], more accurate online optimization or enumeration procedures are proposed, but these also require substantial computational effort. As a consequence, numerical demonstrations of these methods to date have only shown good performance for systems with fewer than 5 states.

1.3.2 Contribution

This work develops a new set-based state estimation algorithm by adapting the DI-based reachability method in Chapter 2 to provide accurate prediction sets using only fast interval computations and adding an efficient and accurate correction algorithm. The prediction step of our algorithm is not quite a direct application of the method in Chapter 2. Instead, we show that output measurements can be used to modify the prediction step in a simple but nontrivial way, leading to significantly tighter prediction

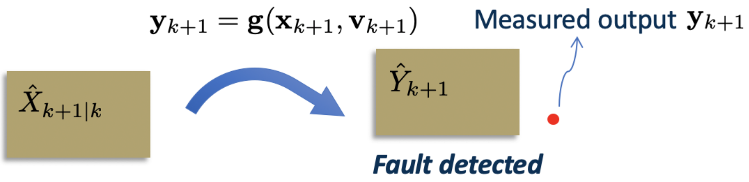
$\mathbf{y}_{k+1} = \mathbf{g}(\mathbf{x}_{k+1}, \mathbf{v}_{k+1})$
Measured output $\mathbf{y}_{k+1}$
$\hat{X}_{k+1|k}$
$\hat{Y}_{k+1}$
Fault detected

Figure 1.3: Fault detection: An interval enclosure of the possible outputs $\hat{Y}_k$ is computed using the prediction set $\hat{X}_{k+1|k}$. A fault is detected when the measured output $\mathbf{y}_{k+1}$ is outside of $\hat{Y}_{k+1}$.

bounds. This method is described in detail in Chapter 3. The numerical results show that this method produces state estimates with significantly higher accuracy and efficiency than state-of-the-art zonotopic methods for a challenging nonlinear chemical reactor model.

1.4 Set-Based Fault Detection

This section introduces Objective 3, which is to apply the efficient and accurate set-based state estimation algorithm developed in Chapter 3 to achieve effective set-based fault detection. To do this, we apply set-based state estimation to the discrete-time dynamic system (1.2) under the assumption that this model describes the nominal dynamics when no fault has occurred. In every time step k, this furnishes a prediction set $\hat{X}_{k+1|k}$ enclosing all states consistent with the nominal model and all measured outputs up to time k. Then, an enclosure $\hat{Y}_k$ of the set of possible outputs is computed using $\hat{X}_{k+1|k}$. If the new measurement $\mathbf{y}_k$ is outside the enclosure $\hat{Y}_k$, the measured output sequence can no longer be explained by the nominal model, and a fault must have occurred. Figure 1.3 illustrates this procedure using an interval enclosure $\hat{Y}_k$ computed using output function (1.2c) and the prediction set $\hat{X}_{k+1|k}$. A fault is detected in this case because the measured output is outside of $\hat{Y}_k$.

1.4.1 Motivation

In applications such as wind energy, industrial robotics, and chemical processing, increases in complexity and automation have made component malfunctions and other abnormal events an ever-present threat to safety and reliability. Industrial statistics show that in addition to explosions and other major accidents, minor accidents happen frequently and may occur on a daily basis, causing injuries, environmental issues, and billions of dollars of losses every year [1]. Moreover, it is impossible to completely rely on humans to detect these abnormal events due to the size and complexity of modern systems. Therefore, computer-based fault detection algorithms have become an essential feature of modern control systems.

This work aims to develop an automated algorithm to detect the occurrence of abnormal events quickly and accurately. Failing to detect faults quickly and accurately can have potentially serious economic, safety, and environmental consequences. In the United States and United Kingdom, the limitations of existing fault detection systems used in the chemical industry have annually cost their economy 20 and 27 billion dollars respectively [38]. Furthermore, in some applications of robotics (e.g., surgery and transportation), system malfunctions directly cause threats to human safety [39, 40]. Therefore, there is an urgent need for algorithms that can achieve early and accurate fault detection, which can effectively mitigate the safety risks associated with abnormal operations, as well as the economic losses caused by off-spec production, maintenance, and downtime.

1.4.2 Existing Methods

Classical FD methods exploit historical data and are well established for various systems [41]. These methods detect faults by comparing observed measurements with previous statistics, which are often effective with sufficient historical data. However, these data-based methods cannot rigorously distinguish faults from system disturbances.

This issue is particularly pronounced when systems have large uncertainties or when there is a lack of high-quality historical data that is relevant to the current operating conditions (e.g., an abnormal disturbance), which leads to false alarms and missed faults. An alternative class of FD methods exploits first-principles process models, which are available at least at the level of individual process units and subsystems in many applications of interest. In model-based approaches, faults are detected by comparing the process outputs that are consistent with the model (under all relevant uncertainties) to the outputs observed from the real process. Specifically, traditional model-based methods detect faults by checking if the difference between the predicted and measured outputs exceeds a threshold. However, the threshold value is usually empirical. Thus, choosing a threshold that minimizes missed faults without generating too many false alarms is challenging. Set-based FD is a particularly useful model-based approach that attempts to address this threshold problem rigorously. In set-based approaches, all uncertainties, disturbances, and measurement noises are assumed to be bounded and set-based computations are used to rigorously test if a new measured output is consistent with the process model given these bounds. This approach eliminates false alarms, but requires accurate set-based computations to achieve high sensitivity to faults, which is challenging.

Many set-based fault detection methods are available for linear systems using computations with intervals [42, 43], polytopes [44], ellipsoids [45], zonotopes [46, 47, 48], and constrained zonotopes [4]. However, testing the consistency of a measured output with a nonlinear model is significantly more difficult.

One possible approach is to solve a nonlinear global optimization problem in each time step to determine if there exists a feasible point in the model that explains the current measurements. Although this would be accurate, it is clearly computationally intractable for most systems. A closely related idea was proposed in [49] for active input design rather than online fault detection.

A second approach is to use set-based parameter estimation. In this approach, measurements are used to compute an enclosure of the set of model parameters that are consistent with the measurements, and a fault is detected when this enclosure has no overlap with a known set of possible parameter values for the fault-free model. The key challenge in this approach is to compute tight enclosures of the feasible parameter set efficiently online. In [50], this is done using interval-based set inversion techniques. However, the computational cost scales exponentially with the number of uncertain parameters. This method is extended to systems with probabilistic noises using a Bayesian framework in [51]. However, this method does not provide rigorous bounds.

A third approach to set-based FD is to apply set-based state estimation. Recall that, in each time step, a set-based state estimator provides a guaranteed enclosure of the set of states consistent with the model, the bounded uncertainties, and all past measurements. This can then be used to compute an enclosure of the possible model outputs, and a fault is declared if the measured output is outside of this set. Note that some methods actually detect faults by computing a set of possible output prediction errors (i.e, residuals) rather than directly computing a set of possible outputs. As discussed in Chapter §1.3, the key challenge for these methods is to compute sufficiently accurate enclosures of the possible fault-free outputs (or residuals) fast enough for online fault detection. The articles [52] and [53] propose set-based FD approaches based on a Luenberger-type set-based state estimators. However, both methods compute rigorous enclosures of the residuals based on linear differential inclusions for the nonlinear observer error dynamics, which is likely to be very conservative for highly nonlinear systems. The article [54] also uses a Luenberger-type set-based state estimator. However, instead of computing a rigorous enclosure of all possible residuals for the fault-free model, they compute a smaller set of residuals based on a prescribed false alarm rate. Thus, this method is not guaranteed to avoid false alarms. Moreover, computing this set of residuals requires the solution of nonlinear chance constrained optimization problems in each time step, which is likely to be intractable for many systems. In order to reduce conservatism and increase efficiency, some approaches use approximate models with simpler structure. In [55], nonlinear models are linearized before constructing the observer, as in the extended Kalman filter. Similarly, the article [56] approximates nonlinear input-output models using a Takagi-Sugeno fuzzy neural network that is linear in the uncertain parameters. Rigorous ellipsoidal [56] and zonotopic [55] enclosures are then computed for the approximate models and used for fault detection. However, these enclosures are not rigorous for the original nonlinear systems and cannot provide guaranteed fault detection. Finally, the article [57] proposes a set-based fault detection method for continuous-time nonlinear systems based on enclosures of the fault-free states computed using advanced reachability techniques based on differential inequalities (DI). Although these reachability methods are very effective, they do not use measurements to refine the predicted enclosures as in a true set-based state estimator. Rather, measurements are only used to test for faults in each time step. This is a serious limitation and is likely to be prohibitive for systems with large uncertainties, where even the exact reachable set can be large.

1.4.3 Contribution

To address these limitations, this Chapter develops a new set-based FD algorithm based on the set-based state estimation algorithm developed in Chapter 3. This algorithm guarantees no false alarms and improves upon the detection speed and fault sensitivity of existing set-based methods due to the superior accuracy and efficiency of our state estimator. The fault detection algorithm is firstly introduced in Section 4.2. The proposed algorithm is then compared with a popular data-based method based on principal component analysis (PCA), a conventional model-based method using the extended Kalman filter (EKF), and four state-of-the-art set-based algorithms. These algorithms are tested for four case studies and various scenarios within each case study, including fault-free cases with normal disturbances, fault-free cases with large persistent disturbances, and cases with various faults. The results show that the proposed set-based algorithm eliminates false alarms and has the highest fault sensitivity among all set-based methods.

1.5 Zonotope Order Reduction

This section introduces work on zonotope order reduction, which is a contribution of this thesis that is separate from but related to fault detection. A zonotope is a convex polytope that can be represented as the image of a unit hypercube under an affine mapping [2]. Specifically, an n-dimensional zonotope Z is described by

$$Z = \{\mathbf{G}\boldsymbol{\xi} + \mathbf{c} : \|\boldsymbol{\xi}\|_\infty \leq 1\}, \tag{1.3}$$

where $\mathbf{c} \in \mathbb{R}^n$ be the *center* of the zonotope and the n_g columns of $\mathbf{G} \in \mathbb{R}^{n \times n_g}$ are the *generators*. The complexity of a zonotope is described by its *order* $o \equiv n_g/n$ [58]. Increasing o makes zonotopes more flexible, but also more cumbersome to do computations with. *Order reduction* refers to the process of bounding a given zonotope Z within another zonotope of lower complexity $Z' \supset Z$.

1.5.1 Motivations and Existing Methods

Since the seminal work of Kühn [2], zonotopes have been widely adopted as an accurate and efficient way to model bounded uncertainties and noises in a variety of control applications, including reachability analysis [2, 58, 59], state estimation [60, 22, 37, 61, 4], hybrid systems verification [62, 63, 64], robust control [65, 33], and fault detection [47, 48, 66, 7]. Zonotopes are significantly more flexible than parallelotopes and ellipsoids, while requiring much less computational effort than general convex polytopes [4]. However, many operations on zonotopes yield results with higher complexity than their arguments [2], which is a serious limitation, particularly for recursive algorithms. Thus, order reduction methods are essential for many control algorithms and can significantly impact their efficiency and performance. For example, inaccurate reduction can lead to overly conservative set-based estimators, and hence to conservative control actions or ineffective fault detection [64, 4].

Order reduction was first addressed in [2] in the context of reachability analysis. The first general purpose method was proposed in [60], followed shortly by a similar method in [58]. These methods are both very efficient. However, while the method in [60] has been overwhelmingly used in the literature [37, 61, 65, 48], there are no available studies comparing their accuracy. A more sophisticated approach was proposed in [63] and shown to be significantly more accurate than the method in [58], but only for a limited set of tests with low-dimensional zonotopes ($n \leq 4$). Moreover, the method in [60] was not compared. Unfortunately, the method in [63] requires a combinatorial search that is problematic in high-dimensions. To address this, another method was recently proposed in [4] that follows the main insights of [63] but eliminates the combinatorial search using an iterative matrix factorization. It was claimed in [4] that the method matches the accuracy of [63] at significantly lower cost. However, because order reduction was not the focus of that article, the method was only described in the appendix, with no theoretical justification and no comparisons.

1.5.2 Contributions

These four existing zonotope order reduction methods are implemented and compared in Chapter 5. This work makes two main contributions. First, the order reduction method by [4] is presented in detail and its validity is rigorously established. Second, a comprehensive comparison of the existing four methods is presented considering both computational cost and overestimation error for a large test set. The effects of problem dimension, initial zonotope order, and reduced zonotope order are also investigated. The results provide valuable guidance for designing set-based estimation and control algorithms that more effectively balance accuracy with computational cost.

1.6 Reachability Analysis for Safety Verification of Autonomous Vehicles

The objective of this work is to verify the safety of autonomous vehicles during path or trajectory tracking using reachability analysis. A vehicle path or trajectory tracking system is a nonlinear closed-loop system, which can be described by the following continuous-time dynamics:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(t, \mathbf{x}(t), \mathbf{w}(t), \mathbf{u}(t)), \\ \mathbf{u}(t) &= \kappa(t, \mathbf{x}(t), \mathbf{w}(t)), \end{aligned} \tag{1.4}$$

where the state variables $\mathbf{x}(t)$ represent the vehicle's positions, velocities, etc., the vector $\mathbf{w}(t)$ represents disturbances and uncertain parameters, and the control inputs $\mathbf{u}(t)$ are computed by a given tracking controller κ. The controller κ is designed to force $\mathbf{x}$ to reach and follow a given reference path or trajectory, which is not shown in (1.4) for brevity.

Let $\mathbf{x}(t; \mathbf{x}_0, \mathbf{w})$ denote the solution of (1.4) for a given initial condition $\mathbf{x}_0$ and disturbance function $\mathbf{w}$. Given admissible sets of initial conditions X_0 and disturbance functions $\mathcal{W}$, the reachable set of (1.4) is defined for every time $t \in [t_0, t_f]$ as

$$\mathcal{R}(t) = \{\mathbf{x}(t; \mathbf{x}_0, \mathbf{w}) : \mathbf{x}_0 \in X_0, \mathbf{w} \in \mathcal{W}\}.$$

For every time t, $\mathcal{R}(t)$ contains all possible states at t under relevant uncertainties. In order to verify the safety of (1.4), all the states in $\mathcal{R}(t)$ should be safe. Therefore, we aim to compute reachability bounds of $\mathcal{R}(t)$ for the closed-loop system (1.4) to ensure the safety of vehicle trajectories.

1.6.1 Motivation and Existing Methods

Path and trajectory tracking is important in automated driving systems for road vehicles, motion planning for autonomous robots, etc. [67, 68]. However, the reference paths and trajectories computed by such systems, which are safe by design, are not followed exactly by the vehicle due to various uncertainties in the vehicle's dynamics and environment (e.g., model parameters, tire slip, wind, measurement noises, etc.). These deviations can lead to collisions or violations of other safety constraints. Therefore, methods for ensuring safety of a vehicle's real trajectory in real time are essential for achieving safe autonomous systems in practice. For example, such methods will be necessary to realize the anticipated safety benefits of autonomous road vehicles that result from eliminating delayed reactions and other human errors [67].

The existing literature on vehicle safety verification addresses several distinct problems based on how the vehicle's control inputs are handled. One class of methods assumes the inputs obey a probability distribution modeling the action of human drivers and aims to compute the likelihood of a collision [69, 70, 71]. These methods are primarily designed to generate warning alarms for human drivers, not for use in automated control systems.

A second class of methods treats the inputs as degrees of freedom and aims to compute either a feedback law or an open-loop input that guarantees safe trajectories [72, 73, 74, 75]. General approaches in this category require the solution of Hamilton-Jacobi-Isaac (HJI) partial differential equations, which is prohibitive because it scales exponentially in the number of states. This is partially addressed by dimension reduction methods in [73], but remains a significant limitation.

A third class of methods considers the simpler problem of verifying safety for a fixed control input specified *a priori*. This input can be specified as either an open-loop input [76, 77, 78, 79] or a fixed feedback law [80, 34, 81, 10, 82, 83, 84]. Although these methods only assess the safety of a given control input rather than synthesizing a safe input, they address a critical subtask that can be used within larger algorithms for synthesizing safe controllers or motion plans. The methods in [76, 77, 79, 83, 84] compute the probability of safety violations by sampling or using stochastic reachable sets. Therefore, these methods cannot make rigorous safety guarantees, which is a drawback in some applications. Moreover, sampling-based methods are computationally demanding for systems with more than a few uncertain quantities, which limits their use for online safety verification. In contrast, the methods in [78, 80, 34, 81, 10, 82] aim to provide rigorous safety guarantees for systems subject to bounded uncertainties using reachability analysis techniques. However, efficiently computing an accurate enclosure of the reachable set of a nonlinear system is a significant challenge. To avoid this, most safety verification approaches use linear models [78, 81] or linearizations of nonlinear models [34, 74, 80]. Unfortunately, verifying safety of a linearized model does not ensure that the original model is safe. To date, the only guaranteed safety verification approach applicable to nonlinear vehicle models is given in [10, 82]. For the example considered in [10], it was shown that this method can verify the safety of a trajectory about 2× faster than the real vehicle traverses the trajectory. While this is promising, there is still a need for significantly more efficient methods to support verification for more complex models and to enable the use of online verification within iterative algorithms for safe controller synthesis. In practice, autonomous vehicles often update their trajectories every few milliseconds [85, 86, 87], so reachability-based verification on a similar time-scale is desirable.

This chapter focuses on the problem of rigorous safety verification for nonlinear vehicle models under a fixed feedback controller. Specifically, given a vehicle 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.

Many methods are available for computing rigorous reachable set enclosures for continuous-time nonlinear systems. However, these methods often exhibit an unworkable compromise between accuracy and computational efficiency, particularly for systems with strong nonlinearities or large uncertainties. The zonotope-based method in [88], which has been applied for safety verification in [82, 10], propagates valid enclosures over discrete time steps using a conservative linearization technique with rigorously bounded linearization errors. Although this method is effective in many cases, the linearization error bound can be conservative for systems with strong nonlinearities. Moreover, high-order zonotopes and/or partitioning may be required to achieve high accuracy, which may become inefficient. Another class of reachability methods propagates valid enclosures over discrete time steps by first constructing a Taylor expansions of the states with respect to time and then computing rigorous bounds on the coefficients and remainder term [89]. Early methods computed these bounds using interval arithmetic, but contemporary methods achieve much higher accuracy using Taylor model arithmetic, which is based on multivariate Taylor expansions with respect to uncertain parameters [90, 91, 92, 93]. However, high accuracy may require high-order Taylor models, which also comes with high computational cost.

A final class of reachability methods is based on the theory of differential inequalities (DI). These methods compute valid 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 [94]. This is very efficient, which is attractive for online verification, but it usually computes very conservative bounds. Several more recent DI methods have addressed this by replacing intervals with polytopes [27], Taylor models [95], or mean value enclosures [96]. These methods produce much tighter bounds than standard DI, but are not as efficient. Another category of DI methods aims to use model redundancy to mitigate the conservatism of the standard interval 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 within iterative refinement algorithms to tighten the bounds continuously as they are propagated forward in time [24, 26, 96]. Importantly, this method can be applied to general nonlinear systems that do not satisfy any known invariants by *manufacturing* invariants [26]. This process involves embedding the system within a higher-dimensional system that obeys invariants by design (see Chapter 6 for details). 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 [24, 26, 96]. However, this approach requires significant problem insight to apply effectively, especially when invariants must be manufactured. To date, successful strategies have only been clearly demonstrated for models that arise from dynamic mass and energy balances, particularly in the chemical engineering domain, where it is relatively straightforward to manufacture simple and effective affine invariants.

1.6.2 Contributions

In this chapter, we demonstrate the application of advanced redundancy-based DI methods to three representative case studies in vehicle path and trajectory tracking. The application of redundancy-based DI to this class of problems is challenging for three primary reasons. First, to the best of our knowledge, the models we consider do not naturally obey any invariants. Moreover, compared to mass and energy balance models, it much more difficult to identify effective manufactured invariants. Second, the presence of a feedback law in these models causes a significant interval dependency problem, which leads to very conservative bounds using interval-based methods if it is not addressed. Both of these challenges are explained in more detail in Chapter 6.

Finally, the vehicle models of interest involve several functions that do not have well-defined interval evaluations, or whose interval evaluations violate Lipschitz regularity conditions that are required by DI-based reachability methods.

To address these issues, we first develop extended interval operations for several functions common in vehicle models and prove Lipschitz regularity. 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. Moreover, we develop highly effective nonlinear manufactured invariants. In all cases, we ultimately obtain reachability bounds that are greatly improved relative to the standard DI method, and appear both accurate and efficient enough to support many online safety verification tasks, although there is clearly still room for improvement. Finally, we conclude with a discussion of lessons learned and general strategies that are likely to be effective for other path and trajectory tracking problems.

1.6.3 Overall Summary

This dissertation is organized as follows. Chapter 2 introduces a discrete-time reachability analysis method using differential inequalities with invariants. The invariants are manufactured based on the special structure in many chemical reaction and separation models. Moreover, strategies to manufacture invariants for autonomous driving systems are proposed in Chapter 6. Next, Chapter 3 extends the reachability method to set-based state estimation by adding an accurate correction step. Then, Chapter 4 proposes a fast and accurate set-based fault detection method based on the state estimation algorithm in Chapter 3. Although the set-based state estimation and fault detection methods developed in this thesis use interval enclosures, many state-of-the-art methods use zonotopic enclosures. A key challenge with these methods is that many operations on zonotopes increase the complexity of the enclosure, and hence also increase the computational cost of future operations. Chapter 5 provides a comprehensive comparison of zonotope order reduction methods for addressing this problem, along with some new results that can be used to improve set-based state estimation and fault detection algorithms using zonotopes. Finally, Chapter 6 applies differential inequalities with manufactured invariants to verify the safety of autonomous vehicle trajectory tracking systems.

CHAPTER 2

ACCURATE UNCERTAINTY PROPAGATION FOR DISCRETE-TIME NONLINEAR SYSTEMS USING DIFFERENTIAL INEQUALITIES WITH MODEL REDUNDANCY

2.1 Introduction

This chapter presents new methods for accurately and efficiently propagating uncertainty through nonlinear discrete-time models in the form of rigorous interval bounds on the set of possible solutions (i.e., the reachable set). Enclosing reachable sets is a critical step in set-based state estimation [3, 4], which is used in a variety of robust control and fault detection algorithms [5, 6, 7, 97]. Reachable sets are also widely used in safety verification, motion planning, design space construction, and many other applications [9, 10, 11].

A number of effective algorithms are available for bounding the reachable sets of discrete-time linear systems [12, 13, 14, 2]. However, for nonlinear systems, computing accurate enclosures remains a significant challenge, especially when enclosures must be computed rapidly online. In essence, propagating an enclosure X_k from time k to $k+1$ is equivalent to bounding the image of X_k under the nonlinear vector function defining the dynamics. This can be done efficiently using interval arithmetic [15], but the resulting enclosure is often very conservative. A tighter interval enclosure can be obtained by partitioning X_k, but this is much more costly [15, 16]. For polynomial systems, tighter bounds on the image of X_k have been obtained using optimization formulations such as linear, semidefinite, and DC programming [17, 18, 19]. In [20], a new approach for efficiently optimizing polynomials over parallelotopes using the Bernstein basis is applied, and in [11, 21], polynomial dynamics are represented as linear fractional transformations and bounded using the skewed structured singular value. However, optimization-based approaches may be too computationally demanding for online applications. A faster approach is to propagate X_k from k to $k+1$ by first considering a local linearization of the dynamics and then adding a rigorous bound on the linearization error [22, 23]. This strategy exploits efficient set-based calculations that are possible in the linear case (e.g., using zonotopes), but can suffer from large linearization errors for nonlinear systems.

In this chapter, we present two new methods for discrete-time uncertainty propagation motivated by continuous-time methods based on differential inequalities (DI) [28]. For continuous-time systems, the basic DI method uses interval arithmetic to construct bounding differential equations that furnish time-varying interval bounds as their solutions [94, 24]. Like discrete-time interval methods (without partitioning), this produces bounds at low cost, but can be very conservative. However, DI is not directly analogous to discrete-time interval methods, and it does provide sharp bounds in many nontrivial cases, such as for quasi-monotone systems [28]. More importantly, several advanced DI methods have recently been developed that largely retain the efficiency of the original method while providing much tighter enclosures [98, 25, 24, 99, 100, 27, 26]. Thus, there is significant motivation to extend these approaches to discrete time.

However, DI theory depends critically on the fact that, for continuous-time systems, a trajectory cannot leave a set X without crossing its boundary. Thus, to propagate a reachable set enclosure forward in time, it is enough to consider the behavior of the vector field on its boundary [28]. Unfortunately, this is not true in discrete-time, and this precludes any straightforward analogue of the DI approach for general discrete-time systems.

Despite this fact, the first main result of this chapter shows that a direct discrete-time extension of the standard DI method in [94] does provide valid reachability bounds whenever the right-hand side of the discrete-time system satisfies a simple monotonicity property. Since this property is not required in continuous-time, it stands to reason that it should hold generally for discrete-time systems that accurately approximate a continuous-time model. Indeed, we show that the required property holds for any system derived by forward Euler discretization provided that the right-hand side is locally Lipschitz continuous and the step size is below an easily computable upper bound. This bound is no more restrictive than that required to preserve basic physical properties of the solution, such as non-negativity [29].

Next, we consider the advanced DI method in [24, 26], which achieves much tighter bounds than standard DI by exploiting redundant model equations within an iterative bound refinement operator. We show that this method is also valid in discrete-time provided that the refinement operator satisfies a Lipschitz condition and the right-hand side function satisfies a stronger monotonicity property that depends on the refinement Lipschitz constant. Again, we show that this property always holds for Euler discretized systems with sufficiently small step size, but the step size limit in this case also depends on the refinement Lipschitz constant. Two new refinement algorithms are then presented that are deliberately designed to balance refinement accuracy with a low Lipschitz constant, and simple formulas are established for computing this constant. Finally, the new DI methods are compared to a standard discrete-time interval method and two popular methods using zonotopes [22, 23]. Our results show that discrete-time DI offers significant advantages in terms of both speed and accuracy for some challenging examples.

We emphasize that our aim is not to bound the reachable sets of continuous-time systems using discrete-time approximations, but rather to bound the reachable sets of discrete-time systems directly for use in discrete-time algorithms for robust estimation and control. Nevertheless, many discrete-time systems of interest are approximations of continuous-time systems, and our results show that this leads to properties that are useful for bounding. Also, note that the methods herein strictly do not use differential inequalities since they are formulated in discrete-time. We refer to them as discrete-time DI methods only to emphasize their direct connection to true DI methods for continuous-time systems.

In practice, however, discrete-time systems are often obtained by forward Euler discretization of continuous-time models. Focusing on this special case, our main results show that, for any given system, there exists a bound on the discretization step size below which a discrete-time analogue of the basic DI method provides bounds on the reachable sets of the discretized system. This step size bound can be easily computed in advance, and is no more restrictive than the step size required to preserve basic physical properties of the solution, such as non-negativity [29]. Moreover, the DI bounding results can be applied to general discrete-time systems as well under a monotonicity condition. Next, we show that the advanced DI methods in [24, 26] are also valid in discrete time under a tighter step size restriction. This theoretical development can be generalized to consider dynamic systems subject to externally imposed state nonlinear constraints, where one is only interested in bounding the feasible trajectories, which is useful in optimal control applications. We compare both methods to the standard discrete-time interval method and two popular methods using zonotopes [22, 23]. Our results show that discrete-time DI offers significant advantages in terms of both speed and accuracy.

The remainder of Section 2.1 gives a formal problem statement and introduces some required notation. Next, Sections 2.2 and 2.3 develop discrete-time extensions of the standard DI method in [94] and the advanced DI method in [24, 26]. Section 2.5 develops new algorithms for bound refinement based on nonlinear constraints. Case studies are presented in Section 2.6 and Section 2.7 gives concluding remarks.

2.1.1 Problem Statement

Let $\mathbb{K} \equiv \{0, \ldots, K\}$ be a time horizon of interest and let $\mathbf{h} : D_h \subset \mathbb{K} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$. Moreover, let $G \subset \mathbb{K} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w}$ and consider the constrained discrete-time system:

$$\mathbf{x}_{k+1} = \mathbf{h}\left(k, \mathbf{x}_k, \mathbf{w}_k\right), \tag{2.1a}$$

$$\mathbf{x}_0 = \mathbf{c}_0, \tag{2.1b}$$

$$(k, \mathbf{x}_k, \mathbf{w}_k) \in G. \tag{2.1c}$$

Let $C_0 \subset \mathbb{R}^{n_x}$ and $W \subset \mathbb{R}^{n_w}$ be intervals of admissible initial conditions and disturbances, respectively. Define the shorthand $\mathbf{x}_{0:K} = (\mathbf{x}_0, \ldots, \mathbf{x}_K)$, $\mathbf{w}_{0:K} = (\mathbf{w}_0, \ldots, \mathbf{w}_K)$, $W_{0:K} \equiv W \times \cdots \times W$, and $\mathbb{R}^{n_x}_{0:K} \equiv \mathbb{R}^{n_x} \times \cdots \times \mathbb{R}^{n_x}$. We call $(\mathbf{c}_0, \mathbf{w}_{0:K}, \mathbf{x}_{0:K}) \in C_0 \times W_{0:K} \times \mathbb{R}^{n_x}_{0:K}$ a solution of (2.1a)–(2.1b) if it satisfies (2.1a)–(2.1b) for all $k \in \{0, \ldots, K-1\}$, and a solution of (2.1) if it also satisfies (2.1c) for all $k \in \{0, \ldots, K\}$.

Definition 1. The *reachable set* of the discrete-time system (2.1) is defined for every $k \in \mathbb{K}$ by

$$\begin{aligned}\mathcal{R}_k \equiv \{\mathbf{z} \in \mathbb{R}^{n_x} : \exists \text{ a solution } (\mathbf{c}_0, \mathbf{w}_{0:K}, \mathbf{x}_{0:K}) \in C_0 \times W_{0:K} \\ \times \mathbb{R}^{n_x}_{0:K} \text{ of (2.1) satisfying } \mathbf{x}_k = \mathbf{z}\}.\end{aligned}$$

We are interested in computing reachable set enclosures in the form of *state bounds*, defined as follows.

Definition 2. Two sequences $\mathbf{x}^L_{0:K}$ and $\mathbf{x}^U_{0:K}$ are called *state bounds* for (2.1) if $\mathcal{R}_k \subset [\mathbf{x}^L_k, \mathbf{x}^U_k]$, $\forall k \in \mathbb{K}$.

Depending on the application, the inputs $\mathbf{w}_k$ may represent disturbances, control inputs, or model uncertainties. Time-invariant uncertainties $\mathbf{p} \in W$ can be modeled by simply setting $\mathbf{w}_k = \mathbf{p}$ in (2.1). This is a special case of our problem formulation in the sense that any state bounds that are valid for all solutions of (2.1) must also be valid for all solutions with $\mathbf{w}_k = \mathbf{p}$, $\forall k \in \mathbb{K}$. Unconstrained systems are also a special case since we may always choose $G = \mathbb{K} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w}$. In optimal and robust control applications, G may include state constraints that must hold for all solutions of interest, but not necessarily for all solutions of (2.1a)–(2.1b). By Definition 2, state bounds are then only required to enclose the feasible solutions. However, our main motivation for including G in (2.1) is to impose constraints that are *redundant* with (2.1a)–(2.1b) (i.e., they are implied by (2.1a)–(2.1b) and therefore satisfied by every solution of (2.1a)–(2.1b)). Examples include the non-negativity of certain states, conservation laws, and more general invariant sets for (2.1a)–(2.1b). In this case, the reachable sets of (2.1) are the same as those of (2.1a)–(2.1b), and state bounds must enclose all solutions of (2.1a)–(2.1b). We will show that such a set G can often be used to refine state bounds at every k, resulting in much sharper bounds than would be obtained by considering only (2.1a)–(2.1b). This use of model redundancy is now well-established for continuous-time systems [98, 24, 100]. Moreover, for systems that do not satisfy any known redundant constraints, a set G can be manufactured by introducing redundant states into the model, often resulting in much sharper bounds [26].

2.1.2 Notation

For $\mathbf{z}^L, \mathbf{z}^U \in \mathbb{R}^n$, let $Z = [\mathbf{z}^L, \mathbf{z}^U]$ denote the compact n-dimensional interval $\{\mathbf{z} \in \mathbb{R}^n : \mathbf{z}^L \leq \mathbf{z} \leq \mathbf{z}^U\}$, and denote the set of all such intervals by $\mathbb{IR}^n$. For $D \subset \mathbb{R}^n$, let $\mathbb{I}D$ denote the set of all intervals Z such that $Z \subset D$. For $Z \in \mathbb{IR}^n$, define the midpoint $\mathrm{mid}(Z) = \frac{1}{2}(\mathbf{z}^U + \mathbf{z}^L)$, width $w(Z) = \|\mathbf{z}^U - \mathbf{z}^L\|_\infty$, magnitude $|Z| = \max\{\|\mathbf{z}\|_\infty : \mathbf{z} \in Z\}$, and *mignitude* $\langle Z \rangle = \min\{\|\mathbf{z}\|_\infty : \mathbf{z} \in Z\}$ [101].

Denote the Hausdorff distance between two compact sets $Z, X \subset \mathbb{R}^n$ by

$$d_H(Z, X) = \max\{\max_{\mathbf{z}\in Z} \min_{\mathbf{x}\in X} \|\mathbf{x} - \mathbf{z}\|_\infty, \max_{\mathbf{x}\in X} \min_{\mathbf{z}\in Z} \|\mathbf{x} - \mathbf{z}\|_\infty\}. \tag{2.2}$$

For $Z, X \in \mathbb{IR}^n$, the Hausdorff distance simplifies to

$$d_H(Z, X) = \max\{\|\mathbf{z}^L - \mathbf{x}^L\|_\infty, \|\mathbf{z}^U - \mathbf{x}^U\|_\infty\}. \tag{2.3}$$

2.2 State Bounds for Unconstrained Systems

This section presents a new method for computing state bounds for the unconstrained system (2.1a)–(2.1b), which is equivalent to (2.1) with $G = \mathbb{K} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w}$. We assume throughout that an inclusion function is available for $\mathbf{h}$.

Assumption 1. Let $H : D_H \subset \mathbb{K} \times \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_w} \to \mathbb{IR}^{n_x}$ satisfy the condition: For any $(k, Z, V) \in D_H$, the set $\{k\} \times Z \times V$ is contained in D_h and

$$H(k, Z, V) \supset \{\mathbf{h}(k, \mathbf{z}, \mathbf{v}) : (\mathbf{z}, \mathbf{v}) \in Z \times V\}.$$

Denote the elements of H by $H_i = [h_i^L, h_i^U]$.

If $\mathbf{h}(k, \cdot, \cdot)$ is a factorable function (i.e., it can be written explicitly in computer code using a standard mathematics library), then $H(k, \cdot, \cdot)$ can be readily computed as its natural interval extension [102].

Given H, it is well known that state bounds for (2.1a)–(2.1b) can be computed as the solutions of the bounding system [16]:

$$\begin{aligned} x_{k+1,i}^L &= h_i^L(k, [\mathbf{x}_k^L, \mathbf{x}_k^U], W), \quad x_{0,i}^L = c_{0,i}^L, \\ x_{k+1,i}^U &= h_i^U(k, [\mathbf{x}_k^L, \mathbf{x}_k^U], W), \quad x_{0,i}^U = c_{0,i}^U, \end{aligned} \tag{2.4}$$

for all $i \in \{1, \ldots, n_x\}$, where $[\mathbf{c}_0^L, \mathbf{c}_0^U] \equiv C_0$. While this method is very computationally efficient, it usually produces very conservative bounds. In contrast, this section establishes conditions under which (2.4) can be refined by only bounding the range of each h_i over particular faces of the current bounding interval $[\mathbf{x}_k^L, \mathbf{x}_k^U]$.

Definition 3. Let $\beta_i^L, \beta_i^U : \mathbb{IR}^{n_x} \to \mathbb{IR}^{n_x}$ for every $i \in \{1, \ldots, n_x\}$ be defined by

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

We now consider the refinement of (2.4):

$$\begin{aligned} x_{k+1,i}^L &= h_i^L(k, \beta_i^L\left([\mathbf{x}_k^L, \mathbf{x}_k^U]\right), W), \quad x_{0,i}^L = c_{0,i}^L, \\ x_{k+1,i}^U &= h_i^U(k, \beta_i^U\left([\mathbf{x}_k^L, \mathbf{x}_k^U]\right), W), \quad x_{0,i}^U = c_{0,i}^U, \end{aligned} \tag{2.5}$$

for all $i \in \{1, \ldots, n_x\}$. The form of (2.5) is motivated by state bounding methods for continuous-time systems based on the theory of differential inequalities. The key observation in these methods is that a continuous-time trajectory cannot leave a bounding interval without first crossing its boundary. Thus, it is only necessary for a bounding approach to account for the possible values of the vector field on the boundaries of $[\mathbf{x}_k^L, \mathbf{x}_k^U]$; see [24] for details. This is known to lead to much tighter bounds for continuous time systems, specifically because $\beta_i^{L/U}\left([\mathbf{x}_k^L, \mathbf{x}_k^U]\right) \subset [\mathbf{x}_k^L, \mathbf{x}_k^U]$ for all i, and so taking the natural interval extension of a function over $\beta_i^{L/U}\left([\mathbf{x}_k^L, \mathbf{x}_k^U]\right)$ instead of $[\mathbf{x}_k^L, \mathbf{x}_k^U]$ often yields tighter bounds (the result can be the same for some simple functions; see Section 2.6.5). However, this key observation is generally not true for discrete-time systems, since it is possible for $\mathbf{x}_k$ to jump directly from the interior of $[\mathbf{x}_k^L, \mathbf{x}_k^U]$ to its exterior in a single discrete time step. Even so, it could be anticipated that (2.5) would be valid for discrete-time systems that are good approximations of a continuous-time model, which is the case in most applications of interest. The following result shows that (2.5) is in fact valid under a monotonicity condition on $\mathbf{h}$ which is likely to hold for discrete approximations of continuous dynamics, as discussed in §2.2.1.

Theorem 1. *Choose any compact set* $\bar{X}$ *such that* $\mathbb{K} \times \bar{X} \times W \subset D_h$. *For every* $i \in \{1, \ldots, n_x\}$, *assume that*

$$h_i(k, \mathbf{x}, \mathbf{w}) - h_i(k, \hat{\mathbf{x}}, \mathbf{w}) \geq 0, \tag{2.6}$$

for every $(k, \mathbf{x}, \mathbf{w}), (k, \hat{\mathbf{x}}, \mathbf{w}) \in \mathbb{K} \times \bar{X} \times W$ *such that* $x_i \geq \hat{x}_i$ *and* $x_j = \hat{x}_j$, $\forall j \neq i$. *Let* $\mathbf{x}^L_{0:K}$ *and* $\mathbf{x}^U_{0:K}$ *be solutions of* (2.5) *and let* K^* *denote the largest integer in* $\mathbb{K}$ *such that* $[\mathbf{x}^L_{k-1}, \mathbf{x}^U_{k-1}] \subset \bar{X}$ *for all* $k \leq K^*$. *If* $(\mathbf{c}_0, \mathbf{w}_{0:K}, \mathbf{x}_{0:K}) \in C_0 \times W_{0:K} \times \mathbb{R}^{n_x}_{0:K}$ *is a solution of* (2.1a)–(2.1b)*, then* $\mathbf{x}_k \in [\mathbf{x}^L_k, \mathbf{x}^U_k]$, $\forall k \in \{0, \ldots, K^*\}$.

Proof. Let $(\mathbf{c}_0, \mathbf{w}_{0:K}, \mathbf{x}_{0:K}) \in C_0 \times W_{0:K} \times \mathbb{R}^{n_x}_{0:K}$ be a solution of (2.1a)–(2.1b). Choose any $k \in \{0, \ldots, K^* - 1\}$. Using the shorthand $X_k \equiv [\mathbf{x}^L_k, \mathbf{x}^U_k]$, we will show that $\mathbf{x}_k \in X_k$ implies $\mathbf{x}_{k+1} \in X_{k+1}$. Since $\mathbf{x}_0 \in X_0$ by (2.5), the result follows by induction.

Suppose $\mathbf{x}_k \in X_k$. Choose any $i \in \{1, \ldots, n_x\}$ and define $\hat{\mathbf{x}}_k$ by setting $\hat{x}_{k,j} = x_{k,j}$ for all $j \neq i$ and $\hat{x}_{k,i} = x^L_{k,i}$. By definition, $\hat{\mathbf{x}}_k \in \beta^L_i(X_k)$. Thus,

$$\begin{aligned} x^L_{k+1,i} - x_{k+1,i} &= h^L_i\left(k, \beta^L_i(X_k), W\right) - h_i(k, \mathbf{x}_k, \mathbf{w}_k), \\ &\leq h_i(k, \hat{\mathbf{x}}_k, \mathbf{w}_k) - h_i(k, \mathbf{x}_k, \mathbf{w}_k). \end{aligned} \tag{2.7}$$

Since $k < K^*$, the choice of K^* implies that $X_k \subset \bar{X}$. It follows that $\hat{\mathbf{x}}_k, \mathbf{x}_k \in \bar{X}$. Since $\hat{x}_{k,i} = x^L_{k,i} < x_{k,i}$ and $x_{k,j} = \hat{x}_{k,j}$, $\forall j \neq i$, by (2.6) we have $x^L_{k+1,i} - x_{k+1,i} \leq 0$. Since the choice of i was arbitrary, $\mathbf{x}^L_{k+1} - \mathbf{x}_{k+1} \leq \mathbf{0}$. The proof that $\mathbf{x}^U_{k+1} - \mathbf{x}_{k+1} \geq \mathbf{0}$ follows similarly. Therefore, $\mathbf{x}_{k+1} \in X_k$. □

If $\mathbb{K} \times \mathbb{R}^{n_x} \times W \subset D_h$ and (2.6) is known to hold with $\bar{X} = \mathbb{R}^{n_x}$, then there is no need to specify $\bar{X}$ and we can set $K^* = K$. Otherwise, Corollary 1 below provides a computationally verifiable test for satisfaction of (2.6) on compact $\bar{X}$. With $\bar{X} \neq \mathbb{R}^{n_x}$, the inclusion $[\mathbf{x}^L_{k-1}, \mathbf{x}^U_{k-1}] \subset \bar{X}$ must be checked at every k during the solution of the bounding system (2.5). If this inclusion fails at k, then (2.5) is no longer valid and (2.4) must be used instead (see Section 2.5.2 for details).

Corollary 1. Choose any compact convex set $\bar{X}$ such that $\mathbb{K} \times \bar{X} \times W \subset D_h$. For every $i \in \{1, \ldots, n_x\}$ and $k \in \mathbb{K}$, assume that h_i is continuously differentiable with respect to x_i and

$$\min_{(\mathbf{x},\mathbf{w}) \in \bar{X} \times W} \frac{\partial h_i}{\partial x_i}(k, \mathbf{x}, \mathbf{w}) \geq 0. \tag{2.8}$$

Let $\mathbf{x}^L_{0:K}$ and $\mathbf{x}^U_{0:K}$ be solutions of (2.5) and let K^* denote the largest integer in $\mathbb{K}$ such that $[\mathbf{x}^L_{k-1}, \mathbf{x}^U_{k-1}] \subset \bar{X}$ for all $k \leq K^*$. If $(\mathbf{c}_0, \mathbf{w}_{0:K}, \mathbf{x}_{0:K}) \in C_0 \times W_{0:K} \times \mathbb{R}^{n_x}_{0:K}$ is a solution of (2.1a)–(2.1b), then $\mathbf{x}_k \in [\mathbf{x}^L_k, \mathbf{x}^U_k]$, $\forall k \in \{0, \ldots, K^*\}$.

Proof. By Theorem 1, it suffices to show that (2.6) holds for every $(k, \mathbf{x}, \mathbf{w}), (k, \hat{\mathbf{x}}, \mathbf{w}) \in \mathbb{K} \times \bar{X} \times W$ such that $x_i \geq \hat{x}_i$ and $x_j = \hat{x}_j$, $\forall j \neq i$. Choose any such points and any $i \in \{1, \ldots, n_x\}$. Since h_i is continuously differentiable w.r.t. x_i and $\bar{X}$ is convex, the Mean Value Theorem furnishes $\tilde{\mathbf{x}} \in \bar{X}$ such that

$$h_i(k, \mathbf{x}, \mathbf{w}) - h_i(k, \hat{\mathbf{x}}, \mathbf{w}) = \frac{\partial h_i}{\partial x_i}(k, \tilde{\mathbf{x}}, \mathbf{w})(x_i - \hat{x}_i). \tag{2.9}$$

Since $x_i - \hat{x}_i \geq 0$ and $\frac{\partial h_i}{\partial x_i}(k, \tilde{\mathbf{x}}, \mathbf{w}) \geq 0$ by (2.8), we have $h_i(k, \mathbf{x}, \mathbf{w}) - h_i(k, \hat{\mathbf{x}}, \mathbf{w}) \geq 0$ as desired. □

2.2.1 Explicit Euler Systems

An important special case where the improved bounding system (2.5) will very often be valid is when the discrete-time system of interest is derived by forward Euler discretization of a continuous-time model with step size $\delta \in \mathbb{R}_+$. In this case,

$$\mathbf{h}(k, \mathbf{x}, \mathbf{w}) = \mathbf{x} + \delta \mathbf{f}(k, \mathbf{x}, \mathbf{w}) \tag{2.10}$$

for some $\mathbf{f} : D_h \to \mathbb{R}^{n_x}$. Notably, the following result does not require any monotonicity properties of $\mathbf{f}$.

Corollary 2. Choose any compact set $\bar{X}$ such that $\mathbb{K} \times \bar{X} \times W \subset D_h$. Assume that $\mathbf{h}$ is given by (2.10) and let $\mathbf{f}$ satisfy the following Lipschitz condition: There exists $M \in \mathbb{R}_+$ such that

$$|f_i\,(k, \mathbf{x}, \mathbf{w}) - f_i\,(k, \hat{\mathbf{x}}, \mathbf{w})\,| \leq M|x_i - \hat{x}_i|, \tag{2.11}$$

for every $i \in \{1, \ldots, n_x\}$, every $k \in \mathbb{K}$, and every $(\mathbf{x}, \hat{\mathbf{x}}, \mathbf{w}) \in \bar{X} \times \bar{X} \times W$ such that $x_j = \hat{x}_j$ for all $j \neq i$. Let $\mathbf{x}^L_{0:K}$ and $\mathbf{x}^U_{0:K}$ be solutions of (2.5) and let K^* denote the largest integer in $\mathbb{K}$ such that $[\mathbf{x}^L_{k-1}, \mathbf{x}^U_{k-1}] \subset \bar{X}$ for all $k \leq K^*$. If $\delta \in (0, \frac{1}{M}]$, then every solution $(\mathbf{c}_0, \mathbf{w}_{0:K}, \mathbf{x}_{0:K}) \in C_0 \times W_{0:K} \times \mathbb{R}^{n_x}_{0:K}$ of (2.1a)–(2.1b) satisfies $\mathbf{x}_k \in [\mathbf{x}^L_k, \mathbf{x}^U_k]$, $\forall k \in \{0, \ldots, K^*\}$.

Proof. By Theorem 1, it suffices to show that (2.6) holds for every $(k, \mathbf{x}, \mathbf{w}), (k, \hat{\mathbf{x}}, \mathbf{w}) \in \mathbb{K} \times \bar{X} \times W$ such that $x_i \geq \hat{x}_i$ and $x_j = \hat{x}_j$, $\forall j \neq i$. For any such points, (2.11) gives

$$\begin{aligned} h_i(k, \hat{\mathbf{x}}, \mathbf{w}) - {} & h_i(k, \mathbf{x}, \mathbf{w}) \\ & = \hat{x}_i - x_i + \delta \left[f_i\,(k, \hat{\mathbf{x}}, \mathbf{w}) - f_i\,(k, \mathbf{x}, \mathbf{w}) \right], \\ & \leq \hat{x}_i - x_i + \delta M \left| \hat{x}_i - x_i \right|, \\ & = (\hat{x}_i - x_i)(1 - \delta M). \end{aligned} \tag{2.12}$$

Since $\delta \in (0, \frac{1}{M}]$ and $\hat{x}_i - x_i \leq 0$, we have $h_i(k, \hat{\mathbf{x}}, \mathbf{w}) - h_i(k, \mathbf{x}, \mathbf{w}) \leq 0$ as desired. □

The step size bound $\delta \leq \frac{1}{M}$ in Corollary 2 is not particularly restrictive. For comparison, if the non-negative orthant is positively invariant for a continuous-time system, then it is necessary to have $\delta \leq \frac{1}{M}$ to ensure non-negativity of the explicit Euler discretization [29]. Therefore, choosing $\delta > \frac{1}{M}$ is unlikely to provide physically meaningful solutions.

If $\mathbb{K} \times \mathbb{R}^{n_x} \times W \subset D_h$ and each f_i is globally Lipschitz continuous w.r.t. x_i, then there exists M satisfying (2.11) without restricting $\mathbf{x}$ and $\hat{\mathbf{x}}$ to a compact set $\bar{X}$. However, if $\mathbf{f}$ is only locally Lipschitz, which is far more common, then M is only guaranteed to exist for $\mathbf{x}, \hat{\mathbf{x}} \in \bar{X}$, and M depends on the choice of $\bar{X}$. This choice is discussed further in Section 2.5.2. Theorem 2 gives a simple means to compute M when $\mathbf{f}$ is continuously differentiable and $\bar{X}$ is an interval.

Theorem 2. *Choose any $\bar{X} \in \mathbb{IR}^{n_x}$, assume that $\mathbf{f}$ is continuously differentiable, and let $\mathbf{J} \in \mathbb{R}^{n_x \times n_x}$ have elements satisfying $J_{ij} \geq \max_{\mathbf{x} \in \bar{X}, \mathbf{w} \in W} |\frac{\partial f_i}{\partial x_j}(\mathbf{x}, \mathbf{w})|$. Then the inequality*

$$\|\mathbf{f}(\mathbf{x}, \mathbf{w}) - \mathbf{f}(\hat{\mathbf{x}}, \mathbf{w})\|_p \leq M \|\mathbf{x} - \hat{\mathbf{x}}\|_p \tag{2.13}$$

holds for all $(\mathbf{x}, \hat{\mathbf{x}}, \mathbf{w}) \in \bar{X} \times \bar{X} \times W$ with $M = \|\mathbf{J}\|_p$, where $\|\cdot\|_p$ denotes the standard p-norm with any $p \geq 1$ or $p = \infty$.

Proof. For any $(\mathbf{x}, \hat{\mathbf{x}}, \mathbf{w}) \in \bar{X} \times \bar{X} \times W$ and any $i \in \{1, \dots, n_x\}$, the Mean Value Theorem gives $\epsilon \in \bar{X}$ such that

$$f_i(\mathbf{x}, \mathbf{w}) - f_i(\hat{\mathbf{x}}, \mathbf{w}) = \frac{\partial f_i}{\partial \mathbf{x}}(\epsilon, \mathbf{w})(\mathbf{x} - \hat{\mathbf{x}}). \tag{2.14}$$

Letting $|\cdot|$ denote the componentwise absolute value,

$$|f_i(\mathbf{x}, \mathbf{w}) - f_i(\hat{\mathbf{x}}, \mathbf{w})| \leq \left|\frac{\partial f_i}{\partial \mathbf{x}}(\epsilon, \mathbf{w})\right| |\mathbf{x} - \hat{\mathbf{x}}|, \tag{2.15}$$

and hence $|\mathbf{f}(\mathbf{x}, \mathbf{w}) - \mathbf{f}(\hat{\mathbf{x}}, \mathbf{w})| \leq \mathbf{J} |\mathbf{x} - \hat{\mathbf{x}}|$. The result follows by taking p-norms on both sides. $\square$

2.3 State Bounds for Constrained Systems

In this section, the bounding method (2.5) is extended to the case where a nontrivial constraint set G is available that can be used to refine the bounds at each k. The results are again motivated by continuous-time methods based on differential inequalities, where it has been shown that the use of G often results in much sharper bounds [24, 100, 26, 103]. In these approaches, the constraint set G is enforced using an interval refinement operator $\mathcal{I}_G$. This operator must return an interval satisfying an inclusion property which, translated to the discrete-time setting, is of the following form, where (Z, V) is an arbitrary interval in $\mathbb{IR}^{n_x} \times \mathbb{IR}^{n_w}$:

$$\mathcal{I}_G(k, Z, V) \supset \{(\mathbf{z}, \mathbf{v}) \in Z \times V : (k, \mathbf{z}, \mathbf{v}) \in G\}. \tag{2.16}$$

In the continuous-time theory, $\mathcal{I}_G$ is also required to satisfy a local Lipschitz property with respect to Z, but the size of the Lipschitz constant is not important. In contrast, the validity of the discrete-time analogue of these methods presented below depends on the size of this constant. Accordingly, it is important here to develop refinement methods with Lipschitz constants that are both small and simple to compute (in order to test the validity of the bounding method for a given system). Notably, however, the analysis below shows that the refinements done by $\mathcal{I}_G$ are only required to be Lipschitz when they are applied to the facets $Z = \beta_i^{L/U}([\mathbf{x}_k^L, \mathbf{x}_k^U])$ appearing in (2.5), as is done in the continuous-time methods [24, 103]. In contrast, there is no such requirement for refinements applied to $Z = [\mathbf{x}_k^L, \mathbf{x}_k^U]$ in every time step. To exploit this observation, we develop the theory below with a generalized operator taking two Z inputs; i.e., $\mathcal{I}_G(k, Z', Z, V)$, and only required to be Lipschitz with respect to Z'. In Section 2.5, this will provide the flexibility to develop specific implementations of $\mathcal{I}_G$ that compromise between the accuracy of the refinement and the resulting Lipschitz constant by replacing $\beta_i^{L/U}([\mathbf{x}_k^L, \mathbf{x}_k^U])$ with $[\mathbf{x}_k^L, \mathbf{x}_k^U]$ in certain places in the algorithm.

The specific requirements on $\mathcal{I}_G$ are given in the following assumption.

Assumption 2. Let $E_{\mathcal{I}} \subset \mathbb{K} \times \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_w}$ and let

$$D_{\mathcal{I}} = \{(k, Z', Z, V) : (k, Z, V) \in E_{\mathcal{I}},\ Z' \in \mathbb{I}Z\}.$$

Let $\mathcal{I}_G : D_{\mathcal{I}} \to \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_w}$ satisfy:

1. For any $(k, Z', Z, V) \in D_{\mathcal{I}}$,

$$\mathcal{I}_G(k, Z', Z, V) \supset \{(\mathbf{z}, \mathbf{v}) \in Z' \times V : (k, \mathbf{z}, \mathbf{v}) \in G\}.$$

2. For any $(k, Z', Z, V) \in D_{\mathcal{I}}$, $\mathcal{I}_G(k, Z', Z, V) \subset Z' \times V$.

3. For any compact $\bar{X} \subset \mathbb{R}^{n_x}$ such that $\mathbb{K} \times \mathbb{I}\bar{X} \times \mathbb{I}W \subset E_{\mathcal{I}}$, $\exists M_{\mathcal{I}} \in \mathbb{R}_+$ such that

$$d_H(\mathcal{I}_G(k, Z', Z, V), \mathcal{I}_G(k, \hat{Z}', Z, V)) \leq M_{\mathcal{I}} d_H(Z', \hat{Z}'),$$

   for all $(k, Z, V) \in \mathbb{K} \times \mathbb{I}\bar{X} \times \mathbb{I}W$ and $Z', \hat{Z}' \in \mathbb{I}Z$.

Remark 1. If G is defined by a set of constraints, e.g., $\mathbf{g}(k, \mathbf{x}, \mathbf{w}) = \mathbf{0}$, then $\mathcal{I}_G$ will often involve evaluating an inclusion function for $\mathbf{g}$ or its derivatives at (k, Z, V) or (k, Z', V) (see Section 2.5). In this case, the set $E_{\mathcal{I}}$ represents the domain of this inclusion function. Many standard methods for refining interval enclosures based on a set of constraints violate Condition 3 (particularly in the way division by zero and other domain violations are handled) or result in large $M_{\mathcal{I}}$. Specific algorithms that satisfy Assumption 2 are developed in §2.5.

We now consider the bounding system

$$x^L_{k+1,i} = h^L_i(k, \mathcal{I}_G[k, \beta^L_i(X_k), X_k, W]), \quad x^L_{0,i} = c^L_{0,i}, \tag{2.17}$$
$$x^U_{k+1,i} = h^U_i(k, \mathcal{I}_G[k, \beta^U_i(X_k), X_k, W]), \quad x^U_{0,i} = c^U_{0,i},$$

for all $i \in \{1, \ldots, n_x\}$, where $X_k = [\mathbf{x}_k^L, \mathbf{x}_k^U]$. The next theorem shows that (2.17) is valid provided that $\mathbf{h}$ satisfies a stronger monotonicity condition that depends on $M_{\mathcal{I}}$.

Theorem 3. *Choose any compact set* $\bar{X}$ *such that* $\mathbb{K} \times \bar{X} \times W \subset D_h$ *and* $\mathbb{K} \times \mathbb{I}\bar{X} \times \mathbb{I}W \subset E_{\mathcal{I}}$*, and let* $M_{\mathcal{I}}$ *satisfy Condition 3 of Assumption 2. For every* $i \in \{1, \ldots, n_x\}$*, assume that*

$$h_i(k, \mathbf{x}, \mathbf{w}) - h_i(k, \hat{\mathbf{x}}, \hat{\mathbf{w}}) \geq 0, \tag{2.18}$$

for every $(k, \mathbf{x}, \mathbf{w}), (k, \hat{\mathbf{x}}, \hat{\mathbf{w}}) \in \mathbb{K} \times \bar{X} \times W$ *such that* $x_i \geq \hat{x}_i$ *and*

$$\|(\mathbf{x}, \mathbf{w}) - (\hat{\mathbf{x}}, \hat{\mathbf{w}})\|_\infty \leq M_{\mathcal{I}}(x_i - \hat{x}_i). \tag{2.19}$$

Let $\mathbf{x}_{0:K}^L$ *and* $\mathbf{x}_{0:K}^U$ *be solutions of* (2.17) *and let* K^* *denote the largest integer in* $\mathbb{K}$ *such that* $[\mathbf{x}_{k-1}^L, \mathbf{x}_{k-1}^U] \subset \bar{X}$ *for all* $k \leq K^*$*. If* $(\mathbf{c}_0, \mathbf{w}_{0:K}, \mathbf{x}_{0:K}) \in C_0 \times W_{0:K} \times \mathbb{R}^{n_x}_{0:K}$ *is a solution of* (2.1)*, then* $\mathbf{x}_k \in [\mathbf{x}_k^L, \mathbf{x}_k^U]$, $\forall k \in \{0, \ldots, K^*\}$.

Proof. Let $(\mathbf{c}_0, \mathbf{w}_{0:K}, \mathbf{x}_{0:K}) \in C_0 \times W_{0:K} \times \mathbb{R}^{n_x}_{0:K}$ be a solution of (2.1). Choose any $k \in \{0, \ldots, K^* - 1\}$. Using the shorthand $X_k \equiv [\mathbf{x}_k^L, \mathbf{x}_k^U]$, we will show that $\mathbf{x}_k \in X_k$ implies $\mathbf{x}_{k+1} \in X_{k+1}$. Since $\mathbf{x}_0 \in X_0$ by (2.17), the result follows by induction.

Choose any $i \in \{1, \ldots, n_x\}$ and define the interval X_k^i by setting $X_{k,j}^i = [x_{k,j}^L, x_{k,j}^U]$ for all $j \neq i$ and $X_{k,i}^i = [x_{k,i}, x_{k,i}]$. Since $k < K^*$, we have $X_k \subset \bar{X}$. Thus, by Condition 3 of Assumption 2, it follows that

$$\begin{aligned} d_H(\mathcal{I}_G[k, X_k^i, X_k, W], &\mathcal{I}_G[k, \beta_i^L(X_k), X_k, W]) \\ &\leq M_{\mathcal{I}} d_H(X_k^i, \beta_i^L(X_k)), \\ &\leq M_{\mathcal{I}}(x_{k,i} - x_{k,i}^L). \end{aligned} \tag{2.20}$$

By the definition of X_k^i, Condition 1 of Assumption 2 ensures that $(\mathbf{x}_k, \mathbf{w}_k) \in$

$\mathcal{I}_G(k, X_k^i, X_k, W)$. Let

$$(\mathbf{x}^*, \mathbf{w}^*) \in \underset{(\mathbf{x},\mathbf{w})\in\mathcal{I}_G(k,\beta_i^L(X_k),X_k,W)}{\operatorname{argmin}} \|(\mathbf{x}, \mathbf{w}) - (\mathbf{x}_k, \mathbf{w}_k)\|_\infty. \tag{2.21}$$

By the definition of d_H in (2.2), it follows from (2.20) that

$$\|(\mathbf{x}^*, \mathbf{w}^*) - (\mathbf{x}_k, \mathbf{w}_k)\|_\infty \leq M_{\mathcal{I}}(x_{k,i} - x_{k,i}^L). \tag{2.22}$$

But Condition 2 of Assumption 2 implies $x_{k,i}^* = x_{k,i}^L$, so

$$\|(\mathbf{x}^*, \mathbf{w}^*) - (\mathbf{x}_k, \mathbf{w}_k)\|_\infty \leq M_{\mathcal{I}}\left(x_{k,i} - x_{k,i}^*\right). \tag{2.23}$$

Since $(\mathbf{x}^*, \mathbf{w}^*) \in \mathcal{I}_G(k, \beta_i^L(X_k), X_k, W)$, Condition 1 of Assumption 1 gives

$$\begin{aligned} & x_{k+1,i}^L - x_{k+1,i} \\ &\quad = h_i^L\left(k, \mathcal{I}_G\left[k, \beta_i^L\left(X_k\right), X_k, W\right]\right) - h_i\left(k, \mathbf{x}_k, \mathbf{w}_k\right) \\ &\quad \leq h_i(k, \mathbf{x}^*, \mathbf{w}^*) - h_i\left(k, \mathbf{x}_k, \mathbf{w}_k\right). \end{aligned} \tag{2.24}$$

In light of (2.23), the assumed monotonicity property of h_i implies that $x_{k+1,i}^L - x_{k+1,i} \leq 0$. Since the choice of i was arbitrary, $\mathbf{x}_{k+1}^L - \mathbf{x}_{k+1} \leq \mathbf{0}$. The proof that $\mathbf{x}_{k+1}^U - \mathbf{x}_{k+1} \geq \mathbf{0}$ follows similarly. □

Corollary 3. Choose any compact convex set $\bar{X}$ such that $\mathbb{K} \times \bar{X} \times W \subset D_h$ and $\mathbb{K} \times \mathbb{I}\bar{X} \times \mathbb{I}W \subset E_{\mathcal{I}}$, and let $M_{\mathcal{I}}$ satisfy Condition 3 of Assumption 2. For every $i \in \{1, \ldots, n_x\}$ and $k \in \mathbb{K}$, assume that $h_i(k, \cdot, \cdot)$ is continuously differentiable and

$$\frac{\partial h_i}{\partial x_i}(k, \eta) - M_{\mathcal{I}}\left(\sum_{j\neq i}^{n_x}\left|\frac{\partial h_i}{\partial x_j}(k, \eta)\right| + \sum_{l=1}^{n_w}\left|\frac{\partial h_i}{\partial w_l}(k, \eta)\right|\right) \geq 0, \tag{2.25}$$

for all $\eta \in \bar{X} \times W$. Let $\mathbf{x}_{0:K}^L$ and $\mathbf{x}_{0:K}^U$ be solutions of (2.17) and let K^* denote the largest integer in $\mathbb{K}$ such that $[\mathbf{x}_{k-1}^L, \mathbf{x}_{k-1}^U] \subset \bar{X}$ for all $k \leq K^*$. If $(\mathbf{c}_0, \mathbf{w}_{0:K}, \mathbf{x}_{0:K}) \in C_0 \times W_{0:K} \times \mathbb{R}_{0:K}^{n_x}$ is a solution of (2.1), then $\mathbf{x}_k \in [\mathbf{x}_k^L, \mathbf{x}_k^U]$, $\forall k \in \{0, \ldots, K^*\}$.

Proof. By Theorem 3, it suffices to show that (2.18) holds for every $(k, \mathbf{x}, \mathbf{w}), (k, \hat{\mathbf{x}}, \hat{\mathbf{w}}) \in \mathbb{K} \times \bar{X} \times W$ satisfying (2.19) and $x_i \geq \hat{x}_i$. Since $h_i(k, \cdot, \cdot)$ is continuously differentiable and $\bar{X} \times W$ is convex, the Mean Value Theorem gives $\eta \in \bar{X} \times W$ such that,

$$\begin{aligned} &h_i(k, \hat{\mathbf{x}}, \hat{\mathbf{w}}) - h_i(k, \mathbf{x}, \mathbf{w}) \\ &\quad = \sum_{j=1}^{n_x} \frac{\partial h_i}{\partial x_j}(k, \eta)(\hat{x}_j - x_j) + \sum_{l=1}^{n_w} \frac{\partial h_i}{\partial w_l}(k, \eta)(\hat{w}_l - w_l). \end{aligned} \tag{2.26}$$

By (2.19), it follows that

$$\begin{aligned} h_i(k, \hat{\mathbf{x}}, \hat{\mathbf{w}}) - h_i(k, \mathbf{x}, \mathbf{w}) &\leq \frac{\partial h_i}{\partial x_i}(k, \eta)(\hat{x}_i - x_i) \\ + \left(\sum_{j \neq i} \left| \frac{\partial h_i}{\partial x_j}(k, \eta) \right| + \sum_{l=1}^{n_w} \left| \frac{\partial h_i}{\partial w_l}(k, \eta) \right| \right) & M_{\mathcal{I}}(x_i - \hat{x}_i). \end{aligned} \tag{2.27}$$

Since $(\hat{x}_i - x_i) \leq 0$, $h_i(k, \hat{\mathbf{x}}, \hat{\mathbf{w}}) - h_i(k, \mathbf{x}, \mathbf{w}) \leq 0$ by (2.25). $\square$

When $M_{\mathcal{I}}$ can be computed (see §2.5) and $\bar{X}$ is an interval, the inequality (2.25) required by Corollary 3 can be readily checked by interval arithmetic. Choosing $\bar{X}$ more generally, e.g. as a zonotope or a polytope, can make (2.25) less restrictive but also more difficult to check.

2.3.1 Explicit Euler Systems

As with the bounding system (2.5), we now show that the refined system (2.17) is always valid for discrete-time systems derived by forward Euler discretization of a continuous-time model with sufficiently small step size $\delta \in \mathbb{R}_+$. The bound on δ here is slightly stronger than in Corollary 2 and depends on $M_{\mathcal{I}}$. But again, $\mathbf{f}$ need not be monotonic.

Corollary 4. Choose any compact set $\bar{X}$ such that $\mathbb{K}\times\bar{X}\times W \subset D_h$ and $\mathbb{K}\times\mathbb{I}\bar{X}\times\mathbb{I}W \subset E_{\mathcal{I}}$, and let $M_{\mathcal{I}}$ satisfy Condition 3 of Assumption 2. Assume $\mathbf{h}$ is given by (2.10) and $\exists M \in \mathbb{R}_+$ such that

$$\|\mathbf{f}(k,\mathbf{x},\mathbf{w}) - \mathbf{f}(k,\hat{\mathbf{x}},\hat{\mathbf{w}})\|_\infty \leq M\|(\mathbf{x},\mathbf{w}) - (\hat{\mathbf{x}},\hat{\mathbf{w}})\|_\infty \tag{2.28}$$

for every $k \in \mathbb{K}$ and $(\mathbf{x},\mathbf{w}),(\hat{\mathbf{x}},\hat{\mathbf{w}}) \in \bar{X}\times W$. Let $\mathbf{x}^L_{0:K}$ and $\mathbf{x}^U_{0:K}$ be solutions of (2.17) and let K^* denote the largest integer in $\mathbb{K}$ such that $[\mathbf{x}^L_{k-1},\mathbf{x}^U_{k-1}] \subset \bar{X}$ for all $k \leq K^*$. If $\delta \in (0,\frac{1}{MM_{\mathcal{I}}}]$ and $(\mathbf{c}_0,\mathbf{w}_{0:K},\mathbf{x}_{0:K}) \in C_0 \times W_{0:K}\times\mathbb{R}^{n_x}_{0:K}$ is a solution of (2.1), then $\mathbf{x}_k \in [\mathbf{x}^L_k,\mathbf{x}^U_k]$, $\forall k \in \{0,\ldots,K^*\}$.

Proof. By Theorem 3, it suffices to show that (2.18) holds for every $(k,\mathbf{x},\mathbf{w}),(k,\hat{\mathbf{x}},\hat{\mathbf{w}}) \in \mathbb{K}\times\bar{X}\times W$ satisfying (2.19) and $x_i \geq \hat{x}_i$. Using (2.28), we have

$$\begin{aligned}
h_i(k,\hat{\mathbf{x}},\hat{\mathbf{w}}) - h_i(k,\mathbf{x},\mathbf{w}) &= \hat{x}_i - x_i + \delta\left[f_i(k,\hat{\mathbf{x}},\hat{\mathbf{w}}) - f_i(k,\mathbf{x},\mathbf{w})\right],\\
&\leq \hat{x}_i - x_i + \delta M\left\|(\hat{\mathbf{x}},\hat{\mathbf{w}}) - (\mathbf{x},\mathbf{w})\right\|_\infty,\\
&\leq \hat{x}_i - x_i + \delta M M_{\mathcal{I}}\left(x_i - \hat{x}_i\right),\\
&= (\hat{x}_i - x_i)(1-\delta M M_{\mathcal{I}}).
\end{aligned} \tag{2.29}$$

Since $\delta \in (0,\frac{1}{MM_{\mathcal{I}}}]$ and $\hat{x}_i - x_i \leq 0$, we have $h_i(k,\hat{\mathbf{x}},\hat{\mathbf{w}}) - h_i(k,\mathbf{x},\mathbf{w}) \leq 0$, as desired. ☐

Remark 2. Since Corollary 4 requires $\mathbf{f}$ to be locally Lipschitz w.r.t. $\mathbf{w}$ as well as $\mathbf{x}$, computing M requires a slight modification of Theorem 2. Specifically, if $\mathbf{J}$ is as in Theorem 2 and $\mathbf{L} \in \mathbb{R}^{n_x\times n_w}$ satisfies $L_{ij} \geq \max_{\mathbf{x}\in\bar{X},\mathbf{w}\in W} |\frac{\partial f_i}{\partial w_j}(\mathbf{x},\mathbf{w})|$, then (2.28) holds with $M = \|[\mathbf{J}\ \mathbf{L}]\|_\infty$.

2.4 Refinement Operators for Linear Constraints

This section discusses an algorithm that can be used to define the refinement operator $\mathcal{I}_G$ such that Assumption 2 holds and an upper bound on the Lipschitz constant $M_{\mathcal{I}}$

can be easily computed. Suppose that $G \equiv \{\mathbf{z} \in X_{\text{nat}} : \mathbf{Mz} = \mathbf{b}\}$, where $\mathbf{M} \in \mathbb{R}^{m \times n_x}$, $\mathbf{b} \in \mathbb{R}^m$, and $X_{\text{nat}} \in \mathbb{IR}^{n_x}$ is an interval of *natural* bounds (i.e., non-negativity). In this case, [26] gives a specific algorithm for $\mathcal{I}_G(Z)$ which refines each Z_j by considering all rearrangements of $\mathbf{Mz} = \mathbf{b}$ of the form $z_j = m_{ij}^{-1}(b_i - \sum_{l \neq j} m_{il} z_l)$ and bounding the right-hand sides over $Z_{l \neq j}$. Algorithm 1 below is a modification designed have smaller M_G (the mid function returns the middle value of its arguments; see [24] for further explanation of the algorithm). The algorithm in [26] gives a more accurate refinement of Z than Algorithm 1 by executing $[z_j^L, z_j^U] \leftarrow [y_j^L, y_j^U]$ after each pass through the inner loop over i. However, this nests all of the refinements in lines 8-9, leading to large M_G. Theorem 4 gives a computable bound on the constant M_G for Algorithm 1.

Algorithm 1 An implementation of $\mathcal{I}_G$ with small M_G

```
1: function IG($\mathbf{z}^L, \mathbf{z}^U, X_{\text{nat}}, \mathbf{M}, \mathbf{b}, \text{tol}$)
2:     $[\mathbf{z}^L, \mathbf{z}^U] \leftarrow [\mathbf{z}^L, \mathbf{z}^U] \cap X_{\text{nat}}$
3:     $[\mathbf{y}^L, \mathbf{y}^U] \leftarrow [\mathbf{z}^L, \mathbf{z}^U]$
4:     for $q = 1$ to $Q$ do
5:         for $j = 1$ to $n_x$ do
6:             for $i = 1$ to $m$ do
7:                 if $|m_{ij}| > \text{tol}$ then
8:                     $\varsigma_{ij} \leftarrow \frac{b_i}{m_{ij}} + \sum_{l \neq j} \min\left(-\frac{m_{il}}{m_{ij}} z_l^L, -\frac{m_{il}}{m_{ij}} z_l^U\right)$
9:                     $\gamma_{ij} \leftarrow \frac{b_i}{m_{ij}} + \sum_{l \neq j} \max\left(-\frac{m_{il}}{m_{ij}} z_l^L, -\frac{m_{il}}{m_{ij}} z_l^U\right)$
10:                    $y_j^L \leftarrow \text{mid}\left(y_j^L, y_j^U, \varsigma_{ij}\right)$
11:                    $y_j^U \leftarrow \text{mid}\left(y_j^L, y_j^U, \gamma_{ij}\right)$
12:                end if
13:            end for
14:        end for
15:        $[\mathbf{z}^L, \mathbf{z}^U] \leftarrow [\mathbf{y}^L, \mathbf{y}^U]$
16:    end for
17:    return $[\mathbf{z}^L, \mathbf{z}^U]$
18: end function
```

Theorem 4. *Define $\mathcal{I}_G$ as in Algorithm 1 and let Q be as in line 4. For each $i \in \{1, \ldots, m\}$, define $m_i^* \equiv \min_j\{|m_{ij}| : |m_{ij}| \geq tol\}$ and $\alpha_i \equiv (\|\mathbf{m}_i\|_1 / m_i^*) - 1$. Definition 4 is satisfied with $M_G = [\max(\max_i \alpha_i, 1)]^Q$.*

Proof. Consider the case $Q = 1$. The intersection in line 2 can be written using min and max operators, which are both Lipschitz with constants 1. Thus, after line 2, the dependence of $\mathbf{z}^L$ and $\mathbf{z}^U$ on the input bounds is Lipschitz with constant 1, and the same is true of $\mathbf{y}^L$ and $\mathbf{y}^U$ after line 3. With $Q = 1$, $\mathbf{z}^L$ and $\mathbf{z}^U$ do not change again until line 15.

For any (i, j) with $|m_{ij}| \geq \text{tol}$, lines 8–9 define ζ_{ij} and γ_{ij} as Lipschitz functions of $(\mathbf{z}^L, \mathbf{z}^U)$, and hence of the input bounds, with constants bounded by α_i. Moreover, since the mid operator is Lipschitz with constant 1, lines 10–11 define y_j^L and y_j^U as Lipschitz functions of ζ_{ij}, γ_{ij}, and the previous values of y_j^L and y_j^U with constant 1. Thus, if $M_G = \max(\max_i \alpha_i, 1)$ is a valid Lipschitz constant for y_j^L and y_j^U with respect to the input bounds immediately before an execution of lines 10–11, then M_G is also valid for y_j^L and y_j^U immediately after. But since $M_G \geq 1$, this holds immediately before the first execution of lines 10–11. Then, by induction, M_G is a valid Lipschitz constant for y_j^L and y_j^U at all stages of the algorithm, and by line 15, the same is true of $\mathbf{z}^L$ and $\mathbf{z}^U$ at termination. Finally, it follows readily from the definition of d_H that Lipschitz continuity of the output bounds with respect to the input bounds is equivalent to Lipschitz continuity in d_H, and with the same constant.

The result for $Q > 1$ follows immediately from the observation that setting $Q > 1$ is equivalent to nesting Q calls of Algorithm 1 with $Q = 1$. $\square$

2.5 Refinement Operators for Nonlinear Constraints

This section extends Section 2.4 by considering the case where G is described in terms of nonlinear equality constraints. Specifically, let $\mathbf{g} : D_g \subset \mathbb{K} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_g}$ and let

$$G \equiv \{(k, \mathbf{x}, \mathbf{w}) \in D_g : \mathbf{g}(k, \mathbf{x}, \mathbf{w}) = \mathbf{0}\}. \tag{2.30}$$

Inequality constraints can be handled by simply adding slack variables (see [103]), but we omit them here for brevity. In this case, $\mathcal{I}_G$ is required to satisfy

$$\mathcal{I}_G(k, Z', Z, V) \supset \{(\mathbf{z}, \mathbf{v}) \in Z' \times V : \mathbf{g}(k, \mathbf{z}, \mathbf{v}) = \mathbf{0}\}, \tag{2.31}$$

for all $(k, Z, V) \in E_{\mathcal{I}}$ and $Z' \in \mathbb{I}Z$. A computationally efficient refinement algorithm for G sets of this type is given in [103] based on the interval Krawczyk method [101]. However, that algorithm can have large $M_{\mathcal{I}}$, which is of no consequence in the continuous-time setting considered in [103], but is restrictive in Theorem 3 and Corollaries 3–4 here. Thus, we follow a similar refinement strategy here, but achieve a smaller and more easily computable Lipschitz constant by using Z rather than Z' in some places in the algorithm, and exploiting the fact that $\mathcal{I}_G$ is only required to be locally Lipschitz w.r.t. Z' (the refinement in [103] takes only one Z argument).

Assumption 3. Assume that $\mathbf{g}(k, \cdot, \cdot)$ is continuously differentiable w.r.t. $\mathbf{y} = (\mathbf{x}, \mathbf{w})$ for every $k \in \mathbb{K}$. Moreover, for every $i \in \{1, \dots, n_g\}$ and $j \in \{1, \dots, n_x + n_w\}$, let $\left[\frac{\partial g_i}{\partial y_j}\right] : D_{[g]} \subset \mathbb{K} \times \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_w} \to \mathbb{IR}$ satisfy the condition: For any $(k, Z, V) \in D_{[g]}$, the set $\{k\} \times Z \times V$ is contained in D_g and

$$\left[\frac{\partial g_i}{\partial y_j}\right](k, Z, V) \supset \left\{\frac{\partial g_i}{\partial y_j}(k, \mathbf{z}, \mathbf{v}) : (\mathbf{z}, \mathbf{v}) \in Z \times V\right\}.$$

The refinement algorithms developed below satisfy Assumption 2 with $E_{\mathcal{I}} = D_{[g]}$ and $D_{\mathcal{I}} = \{(k, Z', Z, V) : (k, Z, V) \in D_{[g]},\ Z' \in \mathbb{I}Z\}$. Choose any $(k, Z', Z, V) \in D_{\mathcal{I}}$ and, for brevity, define $Y = Z \times V$ and $Y' = Z' \times V$. Then, the required inclusion (2.31) is equivalent to

$$\mathcal{I}_G(k, Y', Y) \supset \{\mathbf{y} \in Y' : \mathbf{g}(k, \mathbf{y}) = \mathbf{0}\}. \tag{2.32}$$

To enclose this set, consider any $(k, \mathbf{y}) \in \mathbb{K} \times Y'$ such that $\mathbf{g}(k, \mathbf{y}) = \mathbf{0}$ and choose any $\bar{\mathbf{y}} \in Y'$. Since Y' is convex, for any $i \in \{1, \ldots, n_g\}$, the Mean Value Theorem furnishes $\xi \in Y'$ satisfying

$$0 = g_i(k, \mathbf{y}) = g_i(k, \bar{\mathbf{y}}) + \frac{\partial g_i}{\partial \mathbf{y}}(k, \xi)(\mathbf{y} - \bar{\mathbf{y}}). \tag{2.33}$$

Thus, each interval Y'_j can in principle be refined by rearranging this equation for y_j and then bounding over $Y'_{l \neq j}$. However, this may involve division by intervals containing zero, which would violate the required Lipschitz property of $\mathcal{I}_G$. Instead, we proceed by scaling (2.33) by $\mu \in \mathbb{R}$ and adding $y_j - \bar{y}_j$ to both sides to obtain

$$\begin{aligned} y_j = \bar{y}_j + \mu &\left[g_i(k, \bar{\mathbf{y}}) + \sum_{l \neq j} \frac{\partial g_i}{\partial y_l}(k, \xi)(y_l - \bar{y}_l) \right] \\ &+ \left(1 + \mu \frac{\partial g_i}{\partial y_j}(k, \xi) \right)(y_j - \bar{y}_j). \end{aligned} \tag{2.34}$$

Thus, Y'_j can be refined by the inclusion

$$\begin{aligned} y_j \in \bar{y}_j + \mu &\left[g_i(k, \bar{\mathbf{y}}) + \sum_{l \neq j} \left[\frac{\partial g_i}{\partial y_l} \right](k, Y')(Y'_l - \bar{y}_l) \right] \\ &+ \left(1 + \mu \left[\frac{\partial g_i}{\partial y_j} \right](k, Y') \right)(Y'_j - \bar{y}_j). \end{aligned} \tag{2.35}$$

The algorithm in [26] applies this refinement sequentially for each i and j, and with two choices of μ is designed to minimize the width of the third term in (2.35). If $\left[\frac{\partial g_i}{\partial y_j} \right](Y)$ were a singleton, then choosing $\mu = -1 / \left[\frac{\partial g_i}{\partial y_j} \right](Y)$ would eliminate the term. More generally, we consider choosing $\mu = -1 / \operatorname{mid} \left[\frac{\partial g_i}{\partial y_j} \right](k, Y')$. However, this may result in division by zero. One possible fix for this is to use

$$\mu = -\operatorname{sgn}\left\{ \operatorname{mid} \left[\frac{\partial g_i}{\partial y_j} \right](k, Y') \right\} / \max\left\{ \epsilon, \left| \operatorname{mid} \left[\frac{\partial g_i}{\partial y_j} \right](k, Y') \right| \right\},$$

where $\epsilon > 0$ is a user specified tolerance. However, the sgn function would violate Lipschitz continuity of the refinement operator. Instead, two choices of μ are implemented:

$$\mu^{\pm} = \pm 1/\max\left\{\epsilon, \left|\mathrm{mid}\left[\frac{\partial g_i}{\partial y_j}\right](k, Y')\right|\right\}. \tag{2.36}$$

This ensures that the desired sign is always used.

This algorithm has been shown to satisfy a local Lipschitz condition w.r.t. Y' in [26]. However, the Lipschitz constant is difficult to compute and potentially large. To be precise, choose any $\bar{X} \in \mathbb{IR}^{n_x}$ such that $\mathbb{K} \times \mathbb{I}\bar{X} \times \mathbb{I}W \subset D_{[g]}$ and let $\bar{Y} = \bar{X} \times W$ and $\bar{r} = \frac{1}{2}w(\bar{Y})$. Moreover, let B_g be an upper bound on $\|\mathbf{g}(k, \mathbf{y})\|_\infty$ for all $(k, \mathbf{y}) \in \mathbb{K} \times \bar{Y}$, and suppose that M_g satisfies

$$\|\mathbf{g}(k, \mathbf{y}) - \mathbf{g}(k, \hat{\mathbf{y}})\|_\infty \leq M_g \|\mathbf{y} - \hat{\mathbf{y}}\|_\infty, \tag{2.37}$$

for all $(k, \mathbf{y}, \hat{\mathbf{y}}) \in \mathbb{K} \times \bar{Y} \times \bar{Y}$. Finally, let $B_{[g]il} \geq \left|\left[\frac{\partial g_i}{\partial y_l}\right](k, Y)\right|$ for all $(k, Y) \in \mathbb{K} \times \mathbb{I}\bar{Y}$, and let $M_{[g]il}$ satisfy

$$d_H\left(\left[\frac{\partial g_i}{\partial y_l}\right](k, Y'), \left[\frac{\partial g_i}{\partial y_l}\right](k, \hat{Y}')\right) \leq M_{[g]il} d_H(Y', \hat{Y}'), \tag{2.38}$$

for all $(k, Y', \hat{Y}') \in \mathbb{K} \times \mathbb{I}\bar{Y} \times \mathbb{I}\bar{Y}$. Then, letting $\Psi_{ij}(k, Y')$ denote the right-hand side of (2.35) with $\bar{\mathbf{y}} = \mathrm{mid}(Y')$ and μ as in (2.36), it can be shown that

$$d_H(\Psi_{ij}(k, Y'), \Psi_{ij}(k, \hat{Y}')) \leq \alpha_{ij} d_H(Y', \hat{Y}'), \tag{2.39}$$

for all $(k, Y', \hat{Y}') \in \mathbb{K} \times \mathbb{I}\bar{Y} \times \mathbb{I}\bar{Y}$, with

$$\begin{aligned} \alpha_{ij} = 2 + \epsilon^{-1}&\left[M_g + \sum_l \left(B_{[g]il} + \bar{r} M_{[g]il}\right)\right] \\ &+ \epsilon^{-2} M_{[g]ij}\left[B_g + \bar{r}\sum_l B_{[g]il}\right]. \end{aligned} \tag{2.40}$$

We omit the proof here since it is a straightforward but lengthy application of the results in §2.1 of [101]. The constant M_g arises from the dependence of the reference point on Y', while $M_{[g]il}$ arises from the use of Y' as an argument to $\left[\frac{\partial g_i}{\partial y_l}\right]$. If $\left[\frac{\partial g_i}{\partial y_l}\right](k,\cdot)$ is computed as the natural interval extension of $\frac{\partial g_i}{\partial y_l}(k,\cdot)$, then $M_{[g]il}$ exists by Corollary 2.5.31 in [104] and can be computed using the results in §2.1 of [101], but this requires specialized code and the results can be very conservative. On the other hand, inclusion monotonicity implies that $B_{[g]il}$ can be easily computed by $B_{[g]il} = \left|\left[\frac{\partial g_i}{\partial y_l}\right](k,\bar{Y})\right|$.

Applying the refinement (2.35) sequentially for all (i,j), as is done in [103], results in an overall Lipschitz constant of $M_{\mathcal{I}} = 2\Pi_{i,j}\alpha_{ij}$, which is likely to be unacceptably large for many problems. In contrast, applying (2.35) in parallel for all (i,j) and intersecting the results furnishes weaker bounds but the smaller constant $M_{\mathcal{I}} = \max_{i,j}\alpha_{ij}$, which may be acceptable in some cases but is potentially still quite large.

To address these issues, an alternative approach is to use Y instead of Y' when evaluating each $\left[\frac{\partial g_i}{\partial y_l}\right]$ in (2.35). In this case, $\bar{\mathbf{y}}$ does not need to be in Y', but must be in Y so that $\xi \in Y$ is ensured by the Mean Value Theorem. Moreover, $\mu^{\pm}$ should also be computed with Y in place of Y'. Since $Y \supset Y'$, this will result in weaker bounds on y_j. However, as shown in Theorem 5, the resulting Lipschitz bound is simple to compute and potentially much smaller.

This approach is described in detail in Algorithm 2. To keep the Lipschitz constant small, the refinement (2.35) is applied in parallel rather than sequentially, as discussed above. To this end, the dummy variable $\hat{Y}'$ is used to store the refined bounds, while the value of Y' used in lines 9–10 is never updated. In line 11, $\bar{\cap}$ is the extended intersection defined componentwise by $(X\bar{\cap}Z)_i = [\text{middle}\left(z_i^L, x_i^L, x_i^U\right), \text{middle}\left(z_i^U, x_i^L, x_i^U\right)]$, where middle returns the middle value of three scalar arguments. Note that $X\bar{\cap}Z$ agrees with $X \cap Z$ whenever $X \cap Z$ is non-empty and is a singleton contained in X otherwise. This operation is used here so that $\mathcal{I}_G$ never returns the empty set.

Theorem 5. *Assumption 2 holds with* $E_{\mathcal{I}} = D_{[g]}$, $D_{\mathcal{I}} = \{(k, Z', Z, V) : (k, Z, V) \in$

Algorithm 2 An implementation of $\mathcal{I}_G$

```
1:  function $\mathcal{I}_G(k, Z', Z, V)$
2:      $Y \leftarrow Z \times V$
3:      $Y' \leftarrow Z' \times V$
4:      $\bar{\mathbf{y}} \leftarrow \mathrm{mid}(Y)$
5:      $\hat{Y}' \leftarrow Y'$                                  ▷ Copy $Y'$ to store refinements
6:      for $i = 1$ to $n_g$ do
7:          for $j = 1$ to $n_x + n_w$ do
8:              $\mu^+ \leftarrow 1/\max\left(\epsilon, \left|\mathrm{mid}\left[\frac{\partial g_i}{\partial y_j}\right](k, Y)\right|\right)$
9:              $\Phi \leftarrow g_i(k, \bar{\mathbf{y}}) + \sum_{l \neq j}\left[\frac{\partial g_i}{\partial y_l}\right](k, Y)(Y'_l - \bar{y}_l)$
10:             $\Psi \leftarrow \bar{y}_j + \mu^+\Phi + \left(1 + \mu^+\left[\frac{\partial g_i}{\partial y_j}\right](k, Y)\right)(Y'_j - \bar{y}_j)$
11:             $\hat{Y}'_j \leftarrow \hat{Y}'_j \bar{\cap} \Psi$
12:             Repeat lines 10–11 with $\mu^- \leftarrow -\mu^+$
13:         end for
14:     end for
15:     return $\hat{Y}'$
16: end function
```

$D_{[g]}$, $Z' \in \mathbb{I}Z\}$, *and* $\mathcal{I}_G : D_{\mathcal{I}} \to \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_w}$ *defined by Algorithm 2. In particular, choose any compact* $\bar{X} \subset \mathbb{R}^{n_x}$, *let* $\bar{Y} = \bar{X} \times W$, *and let* $B_{[g]il} \geq \left|\left[\frac{\partial g_i}{\partial y_l}\right](k, Y)\right|$ *for all* $(k, Y) \in \mathbb{K} \times \mathbb{I}\bar{Y}$. *Then Condition 3 of Assumption 2 holds with*

$$M_{\mathcal{I}} = 1 + \epsilon^{-1} \max_i \sum_l B_{[g]il}. \tag{2.41}$$

Proof. Condition 1 of Assumption 2 follows directly from (2.35) and the fact that $Y' \subset Y$. Condition 2 follows from the use of $\bar{\cap}$ in line (11), which ensures that $\hat{Y}'$ only ever becomes smaller. To verify Condition 3, choose any compact $\bar{X} \subset \mathbb{R}^{n_x}$ and let $\bar{Y} = \bar{X} \times W$. It is convenient to first consider Algorithm 2 from line 4 to the end, viewed as a function of Y and Y'. Denoting the output by $\mathcal{I}_G(k, Y', Y)$, we will argue that

$$d_H(\mathcal{I}_G(k, Y', Y), \mathcal{I}_G(k, \hat{Y}', Y)) \leq M_{\mathcal{I}} d_H(Y', \hat{Y}'), \tag{2.42}$$

for all $(k, Y) \in \mathbb{K} \times \mathbb{I}\bar{Y}$ and $Y', \hat{Y}' \in \mathbb{I}Y$. From the definitions of Y and Y' in lines

2–3, it then follows that

$$\begin{aligned} d_H(\mathcal{I}_G(k,Z',Z,V),\mathcal{I}_G(k,\hat{Z}',Z,V)) &\leq M_{\mathcal{I}} d_H(Z'\times V,\hat{Z}'\times V), \\ &= M_{\mathcal{I}} d_H(Z',\hat{Z}'), \end{aligned}$$

for all $(k,Z,V)\in\mathbb{K}\times\mathbb{I}\bar{X}\times\mathbb{I}W$ and $Z',\hat{Z}'\in\mathbb{I}Z$, as desired.

To establish (2.42) note that, in every iteration of Algorithm 2, lines 9 and 10 define Ψ as a function of the form $d+\sum_l A_l(Y_l'-\bar{y}_l)$, with scalar d and intervals A_l that are independent of Y'. It follows from §2.1 of [101] that Ψ is therefore Lipschitz continuous with respect to Y' on $\mathbb{I}\bar{Y}$ with the Lipschitz constant $\alpha=\sum_l |A_l|$. Specifically, we have

$$|A_l| = \left|\mu^{\pm}\left[\frac{\partial g_i}{\partial y_l}\right](k,Y)\right| \leq \epsilon^{-1}B_{[g]il}, \quad \forall l\neq j, \tag{2.43}$$

$$|A_j| = \left|\left(1+\mu^{+}\left[\frac{\partial g_i}{\partial y_j}\right](k,Y)\right)\right| \leq 1+\epsilon^{-1}B_{[g]ij}. \tag{2.44}$$

Therefore, $\alpha\leq 1+\epsilon^{-1}\sum_l B_{[g]il}\leq M_{\mathcal{I}}$. The extended intersection in line 11 is Lipschitz with constant 1 (Lemma 2.8 in [105]). Thus, if $\hat{Y}_j$ is Lipchitz on $\mathbb{I}\bar{Y}$ with constant at most $M_{\mathcal{I}}$ prior to the execution of line 11, then the same is true immediately after line 11. But since line 5 trivially defines $\hat{Y}_j$ as a Lipschitz function of Y' with constant $1\leq M_{\mathcal{I}}$ prior to the first execution of line 11, it follows by induction that the output of Algorithm 2 is Lipschitz w.r.t. Y' on $\mathbb{I}\bar{Y}$ with constant at most $M_{\mathcal{I}}$. Therefore (2.42) holds. □

Remark 3. Lines 8–12 in Algorithm 2 can alternatively be replaced by (recall the notation $\langle\cdot\rangle$ from §2.1.2):

```
 8: if $\left\langle\left[\frac{\partial g_i}{\partial y_j}\right](k,Y)\right\rangle \geq \epsilon$ then
 9:     $\Phi \leftarrow g_i(k,\bar{\mathbf{y}})+\sum_{l\neq j}\left[\frac{\partial g_i}{\partial y_l}\right](k,Y)(Y_l'-\bar{y}_l)$
10:     $\Psi \leftarrow \bar{y}_j+\Phi/\left[\frac{\partial g_i}{\partial y_j}\right](k,Y)$
```

```
11:     $\hat{Y}'_j \leftarrow \hat{Y}'_j \bar{\cap} \Psi$
12: end if
```

Letting $S(k, Y)$ denote the set of all pairs (i, j) such that $\left\langle \left[\frac{\partial g_i}{\partial y_j}\right](k, Y) \right\rangle \geq \epsilon$, the resulting Lipschitz constant satisfies

$$\begin{aligned} M_{\mathcal{I}} &\leq \max \left\{ 1, \max_{(k,Y) \in \mathbb{K} \times \mathbb{I}\bar{Y}} \max_{(i,j) \in S(k,Y)} \left\{ \frac{\sum_{l \neq j} B_{[g]il}}{\left\langle \left[\frac{\partial g_i}{\partial y_j}\right](k, Y) \right\rangle} \right\} \right\}, \\ &\leq \max \left\{ 1, \epsilon^{-1} \max_{\substack{i \in \{1,\ldots,n_x\} \\ j \in \{1,\ldots,n_g\}}} \sum_{l \neq j} B_{[g]il} \right\}, \end{aligned} \tag{2.45}$$

which is smaller than (2.41). As with the Krawcyzk-type refinement in Algorithm 2, Y can be replaced by Y' nearly everywhere above at the expense of a larger $M_{\mathcal{I}}$. However, this approach must use Y in line 8, since otherwise the **if** statement would introduce a non-Lipschitz dependence on Y'.

2.5.1 Quadratic Constraints

Section 2.4 presents an alternative refinement algorithm that results in both tighter enclosures and a smaller Lipschitz constant than Algorithm 2 for the case where each g_i is affine. In this section, we present a similar algorithm for the common case where each g_i has a quadratic form

$$g_i(k, \mathbf{z}, \mathbf{v}) = \mathbf{v}^{\mathrm{T}} \mathbf{Q}_i \mathbf{z} + \gamma_i^{\mathrm{T}} \mathbf{z} + \sigma_i^{\mathrm{T}} \mathbf{v} - c. \tag{2.46}$$

Such constraints can be factored in two ways; namely:

$$\underbrace{(\mathbf{v}^{\mathrm{T}} \mathbf{Q}_i + \gamma_i^{\mathrm{T}})}_{\equiv\, \mathbf{m}_i^{\mathrm{T}}(\mathbf{v})} \mathbf{z} = \underbrace{c - \sigma_i^{\mathrm{T}} \mathbf{v}}_{\equiv\, b_i(\mathbf{v})}, \quad \underbrace{(\mathbf{z}^{\mathrm{T}} \mathbf{Q}_i^{\mathrm{T}} + \sigma_i^{\mathrm{T}})}_{\equiv\, \mathbf{n}_i^{\mathrm{T}}(\mathbf{z})} \mathbf{v} = \underbrace{c - \gamma_i^{\mathrm{T}} \mathbf{z}}_{\equiv\, d_i(\mathbf{z})} \tag{2.47}$$

Refining Z' and V by directly rearranging these equations often results in sharper bounds than Algorithm 2, or the alternative approach in Remark 3, because it avoids unnecessarily introducing a reference point $\bar{\mathbf{y}}$. Therefore, using square brackets to denote interval extensions of the quantities in (2.47), we replace lines 7–13 in Algorithm 2 by:

```
 7: for $j = 1$ to $n_x$ do
 8:     if $\langle [m_{ij}]\,(V) \rangle \geq \epsilon$ then
 9:         $\Psi \leftarrow \left( [b_i]\,(V) - \sum_{l \neq j} [m_{il}]\,(V) Z'_l \right) / [m_{ij}](V)$.
10:         $\hat{Y}'_j \leftarrow \hat{Y}'_j \bar{\cap} \Psi$
11:     end if
12: end for
13: for $q = 1$ to $n_w$ do
14:     if $\langle [n_{iq}]\,(Z) \rangle \geq \epsilon$ then
15:         $\Psi \leftarrow \left( [d_i]\,(Z') - \sum_{l \neq q} [n_{il}]\,(Z) V_l \right) / [n_{iq}](Z)$.
16:         $\hat{Y}'_{n_x+q} \leftarrow \hat{Y}'_{n_x+q} \bar{\cap} \Psi$
17:     end if
18: end for
```

Let $S(V)$ and $S'(Z)$ be the sets of all pairs (i,j) such that $\langle [m_{ij}](V) \rangle \geq \epsilon$ and $\langle [n_{ij}](Z) \rangle \geq \epsilon$, respectively. Then, applying the rules in §2.1 of [101], the Lipschitz constant of the updates in lines 7–12 is bounded by

$$M_{\mathcal{I}Z} \leq \max \left\{ 1, \max_{V \in \mathbb{I}W} \max_{(i,j) \in S(V)} \left\{ \frac{\sum_{l \neq j} |[m_{il}](V)|}{\langle [m_{ij}](V) \rangle} \right\} \right\}, \tag{2.48}$$

$$\leq \max \left\{ 1, \epsilon^{-1} \max_{i,j} \sum_{l \neq j} |[m_{il}](W)| \right\}. \tag{2.49}$$

Replacing $[d_i](Z')$ in line 15 by $c - \gamma_i^T Z'$, the Lipschitz constant of the refinements in lines 13–18 is bounded by

$$M_{\mathcal{I}V} \leq \max\left\{1, \max_{Z\in\mathbb{I}\bar{X}} \max_{(i,j)\in S'(Z)} \left(\frac{\|\gamma_i\|_1}{\langle [n_{ij}](Z)\rangle}\right)\right\}, \tag{2.50}$$

$$\leq \max\left\{1, \epsilon^{-1}\max_i \|\gamma_i\|_1\right\}. \tag{2.51}$$

Thus, the Lipschitz constant of the modified algorithm is

$$M_{\mathcal{I}} \leq \max\left\{M_{\mathcal{I}Z}, M_{\mathcal{I}V}\right\}. \tag{2.52}$$

As in Remark 3, Z can be replaced by Z' nearly everywhere in the code above at the expense of a larger $M_{\mathcal{I}}$. However, Z must be used in the **if** statement on line 14 to retain Lipschitz dependence on Z'.

Remark 4. In several experiments we found that lines 13–18 above were only marginally effective compared to lines 7–12, while including these lines significantly increased $M_{\mathcal{I}}$. Thus, another very useful option in practice is to omit lines 13–18, resulting in the smaller Lipschitz bound $M_{\mathcal{I}} \leq M_{\mathcal{I}Z}$.

2.5.2 Implementation Details

This section outlines the entire procedure used to apply the bounding results of this chapter, including guidance on choosing a refinement algorithm and setting the parameters $\bar{X}$, ϵ, M, and $M_{\mathcal{I}}$. The first step is to choose $\bar{X}$. This interval does not directly affect the bounds computed by either of the new methods (2.5) and (2.17). Rather, it is used to test the validity of these methods for a given system. Specifically, $\bar{X}$ affects M and $M_{\mathcal{I}}$, which appear in the monotonicity/step-size conditions in Theorem 1, Theorem 3, and Corollaries 1–4. Moreover, these results require that $X_k \subset \bar{X}$ in order to propagate bounds beyond time k. Thus, $\bar{X}$ should be chosen large enough to contain all trajectories of (2.1) with high likelihood. There is no need for $\bar{X}$ to be a tight enclosure, but choosing a very large interval can lead to overly restrictive monotonicity/step-size conditions. Thus, while $\bar{X}$ does not directly affect the bounds, it does constrain the subsequent choices of parameters that do, as discussed below. In many cases, the physics of the problem suggest reasonable bounds for $\bar{X}$ (e.g., non-negativity). Otherwise, one can perform a small number of simulations and choose $\bar{X}$ to enclose the solutions with some margin for error. Another option is to first compute weak bounds X'_k using (2.4) and then choose $\bar{X}$ as the interval hull of $\cup_{k\in\mathbb{K}} X'_k$.

For explicit Euler systems, the next step is to compute M. This is done using Theorem 2 for problems without constraints and Remark 2 for problems with constraints.

For problems with constraints, the next step is to choose a refinement algorithm $\mathcal{I}_G$ and determine $M_{\mathcal{I}}$. For linear constraints, $\mathcal{I}_G$ and $M_{\mathcal{I}}$ are given by Algorithm 1 and Theorem 4 in [106]. For general nonlinear constraints, $\mathcal{I}_G$ can be defined by Algorithm 2 herein or the modification in Remark 3, with $M_{\mathcal{I}}$ obtained from (2.41) or (2.45), respectively. It is presently unclear if one of these algorithms is preferable in general, and a detailed comparisons is beyond the scope of this work. For quadratic constraints, $\mathcal{I}_G$ is defined by Algorithm 2 with lines 7–13 replaced by the pseudocode given in Section 2.5.1 and $M_{\mathcal{I}}$ is obtained from (2.49), (2.51), and (2.52). In some cases, a smaller $M_{\mathcal{I}}$ can be computed using (2.48) and (2.50) instead of (2.49) and (2.51) , as is done for two examples in Section 2.6. If the resulting $M_{\mathcal{I}}$ is still unacceptably large, then the modification in Remark 4 is used instead.

All of the refinement algorithms require a tolerance $\epsilon > 0$. Choosing ϵ too small can result in large $M_{\mathcal{I}}$, while choosing ϵ too large can make $\mathcal{I}_G$ less effective. For general discrete-time systems or Euler systems with a pre-specified step size δ, the provided bound for $M_{\mathcal{I}}$ can be substituted into the monotonicity/step-size requirements in Corollaries 3–4 to back calculate the smallest ϵ for which the method (2.17) is valid.

Since $M_{\mathcal{I}}$ is always at least 1, there may not exist an admissible ϵ, in which case (2.17) cannot be used. Otherwise, the smallest admissible ϵ corresponds to the most aggressive refinement for which (2.17) is valid. For Euler systems with some latitude in the choice of δ, ϵ should be chosen to enhance the efficacy of $\mathcal{I}_G$. A good heuristic is that ϵ should be less than $\left\langle \left[\frac{\partial g_i}{\partial y_j}\right](k, \bar{Y})\right\rangle$ for at least some (i, j), which is most easily seen in the **if** statements in Remark 3 and Section 2.5.1, although the effect on Algorithm 2 is similar.

Once $\bar{X}$, ϵ, M, and $M_{\mathcal{I}}$ have been computed, the monotonicity/step-size conditions must be checked. To apply the method (2.5), either (2.8) or the condition $\delta \in (0, \frac{1}{M}]$ must hold (for general or Euler systems, resp.). To apply the method (2.17), either (2.25) or $\delta \in (0, \frac{1}{MM_{\mathcal{I}}}]$ must hold. The corresponding method can then be applied until the terminal time K or until $X_k \subset \bar{X}$ fails. The latter case is unlikely with a good choice of $\bar{X}$ and did not occur in any of our numerical experiments. However, if it does occur, then valid bounds can still be propagated to K using (2.4). If $X_{k+j} \subset \bar{X}$ at some later time, then the original method can be resumed, but this is unlikely because (2.4) is typically very conservative. Alternatively, one can attempt to re-establish the validity of the original method at k by choosing a new $\bar{X}$ and repeating the steps above.

2.6 Numerical Results

We compare the performance of five discrete-time reachability methods: (i) the standard interval method (2.4); (ii) a similar method using zonotopes from [23]; (iii) an alternative zonotope method from [22]; (iv) the discrete-time DI method (2.5); and (v) the discrete-time DI method with constraints (2.17). The articles [23, 22] specifically address state estimation rather than reachability, and [22] considers continuous-time systems. However, both provide methods for bounding the image of a zonotope X_k under a nonlinear function. Methods (ii) and (iii) apply these techniques to the right-hand side of (2.1a) to recursively compute zonotopic enclosures of the solutions of (2.1a)–(2.1b). These methods are analogous to Method (i), but use more complex sets. Both methods are implemented with 10^{th} order zonotopes using the order reduction method in [22]. All interval methods use the natural interval extension of $\mathbf{h}$ for H. Method (v) is the only method that makes use of a constraint set G. In all examples, we use only constraints that hold for all solutions of (2.1a)–(2.1b). Thus, the reachable set of the constrained system (2.1) coincides with that of (2.1a)–(2.1b), so all methods are solving the same problem. All methods are compared in terms of the volume or radius of the computed enclosures, the upper and lower bounds for selected states (calculated by projection for zonotopic methods), and the wall clock time per step (MATLAB R2015a on a Dell Precision T1700 with an i5-4690 CPU @ 3.50GHz and 16.0 GB RAM).

2.6.1 Example 1

The following dynamics describe an enzymatic reaction network with six chemical species, where x_i is the concentration (M) of species i [24]:

$$\begin{aligned}
x_{1,k+1} &= x_{1,k} + \delta\left[-k_{1,k}x_{1,k}x_{2,k} + k_{2,k}x_{3,k} + k_{6,k}x_{6,k}\right] \\
x_{2,k+1} &= x_{2,k} + \delta\left[-k_{1,k}x_{1,k}x_{2,k} + k_{2,k}x_{3,k} + k_{3,k}x_{3,k}\right] \\
x_{3,k+1} &= x_{3,k} + \delta\left[k_{1,k}x_{1,k}x_{2,k} - k_{2,k}x_{3,k} - k_{3,k}x_{3,k}\right] \\
x_{4,k+1} &= x_{4,k} + \delta\left[k_{3,k}x_{3,k} - k_{4,k}x_{4,k}x_{5,k} + k_{5,k}x_{6,k}\right] \\
x_{5,k+1} &= x_{5,k} + \delta\left[-k_{4,k}x_{4,k}x_{5,k} + k_{5,k}x_{6,k} + k_{6,k}x_{6,k}\right] \\
x_{6,k+1} &= x_{6,k} + \delta\left[k_{4,k}x_{4,k}x_{5,k} - k_{5,k}x_{6,k} - k_{6,k}x_{6,k}\right]
\end{aligned} \tag{2.53}$$

The parameters $\mathbf{k} = (k_1, \ldots, k_6)$ are uncertain with $\mathbf{k}_k \in W \equiv [\hat{\mathbf{k}}, 10\hat{\mathbf{k}}]$ and $\hat{\mathbf{k}} = (0.1, 0.033, 16, 5, 0.5, 0.3)$. The initial condition is $\mathbf{c}_0 = (34, 20, 0, 0, 16, 0)$ and $C_0 = [\mathbf{c}_0, \mathbf{c}_0]$.

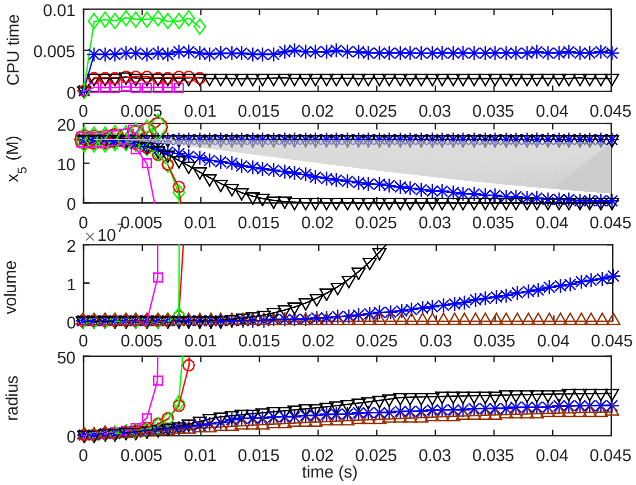

Figure 2.1: Example 2.6.1: Time, state bounds, enclosure volume, and enclosure radius for Methods (i)–(v) (□, ○,◇, ∇, ⋆) and the volume and radius of Method (v) bounds intersected with G (△). Sampled solutions are gray.

Eq. (2.53) satisfies 3 affine solution invariants [24], leading to an *a priori* enclosure $G \equiv \{\mathbf{z} \in X_{\text{nat}} : \mathbf{M}\mathbf{z} = \mathbf{b}\}$ with

$$\mathbf{M} = \begin{bmatrix} 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 1 & -1 & 0 & 1 & -1 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -20 \\ -16 \\ -2 \end{bmatrix}, \tag{2.54}$$

and $X_{\text{nat}} = [0,34] \times [0,20] \times [0,20] \times [0,34] \times [0,16] \times [0,16]$. By Theorem 4, the Lipschitz constant for $\mathcal{I}_G$ defined as in Algorithm 1 with $Q = 1$ is $M_G = 3$. Moreover, choosing $\bar{X} = X_{\text{nat}}$, Theorem 2 bounds the Lipschitz constant for $\mathbf{f}$ by $M = 2665$. We choose $\delta = 9 \times 10^{-5} \leq \frac{1}{MM_G}$ and $K = 500$.

Figure 2.1 shows that the standard interval method (i) produces very weak bounds, as expected. Moreover, using high-order zonotopes in place of intervals, as in Methods (ii)–(iii), is only marginally more effective due to the large linearization errors in this problem. On the other hand, the basic DI method (iv) gives a very significant improvement at low cost, and exploiting G in Method (v) is even more effective, while still requiring less than 10^{-2}s per time step.

2.6.2 Example 2

The following dynamics describe a sewer system with three tanks, where x_i is the water volume (m^3) of tank i and u_i is the i^{th} inlet flowrate (m^3/s) [8]:

$$\begin{aligned} x_{1,k+1} &= x_{1,k} + \delta\left[u_{1,k} + u_{2,k} - \kappa_1 x_{1,k}\right] \\ x_{2,k+1} &= x_{2,k} + \delta\left[\kappa_1 x_{1,k} - \kappa_2\sqrt{x_{2,k}}\right] \\ x_{3,k+1} &= x_{3,k} + \delta\left[\kappa_2\sqrt{x_{2,k}} + u_{3,k} - \kappa_3 x_{3,k}\right] \end{aligned} \tag{2.55}$$

We set $u_i = d_i + w_i$ with $\mathbf{d} = (1, 2, 1)$ and disturbances $w_i \in [0, 0.1]$. The parameters κ_i are also uncertain with bounds $\kappa_1 \in [4.8, 6.8] \times 10^{-4}$, $\kappa_2 \in [1.99, 2.01] \times 10^{-2}$, and $\kappa_3 \in [9.9, 10.1] \times 10^{-4}$. The initial condition is $\mathbf{c}_0 = (167, 1, 333)$ and is certain. Aside from non-negativity of the states, an *a priori* enclosure for (2.55) is not known. Thus, to apply Method (v), we follow the approach in [26] to manufacture a set G by defining the redundant state variable

$$z_k = x_{1,k} + x_{2,k} + x_{3,k}, \tag{2.56}$$

and augmenting (2.55) with the redundant difference equation

$$z_{k+1} = z_k + \delta\left[u_{1,k} + u_{2,k} + u_{3,k} - \kappa_3 x_{3,k}\right]. \tag{2.57}$$

This z_k is chosen so that several terms cancel out when forming (2.57) from (2.55), enabling z_k to be bounded accurately (see [26] for details). Method (v) is then applied to the lifted system consisting of (2.55) and (2.57), which by design satisfies the *a priori* enclosure $G \equiv \{(\mathbf{x}, z) \in \mathbb{R}^4 : z = x_1 + x_2 + x_3,\ (\mathbf{x}, z) \geq \mathbf{0}\}$. By Theorem 4, $M_G = 3$. Choosing $\bar{X} = [167, 7 \times 10^3] \times [1, 5 \times 10^4] \times [333, 8 \times 10^3]$, Theorem 2 gives $M = 0.0111$. We choose $\delta = 30\text{s} \leq \frac{1}{MM_G}$and $K = 600$.

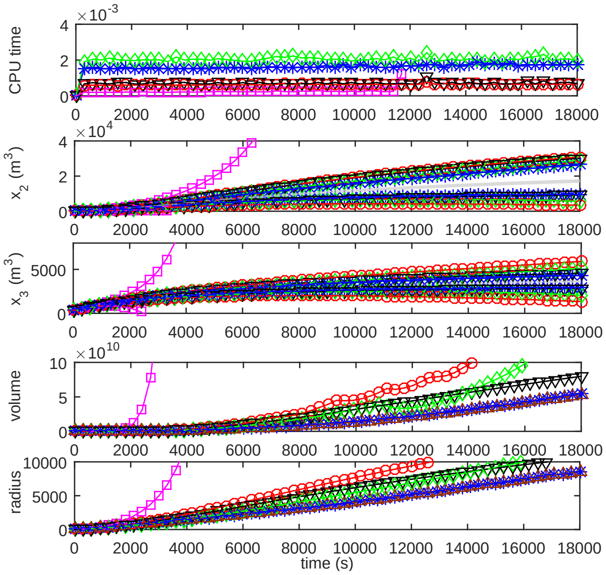

Figure 2.2: Example 2.6.2: Time, state bounds, enclosure volume, and enclosure radius for Methods (i)–(v) (□, ○,◇, ▽, ⋆) and the volume and radius of Method (v) bounds intersected with G (△). Sampled solutions are gray.

Figure 2.2 shows that Method (i) is again very conservative. However, in this case, using zonotopes in Methods (ii)–(iii) leads to a major improvement with modest additional cost. Yet, interval bounds from the simple DI method (iv) are significantly tighter than the zonotopic enclosures, especially at long times. Finally, the use of G again provides the tightest enclosures while retaining high efficiency.

2.6.3 Example 3

The following system results from forward Euler discretization of a continuous-time model of a four species stirred-tank reactor from [27], where x_i is the concentration of species i:

$$\begin{aligned}
x_{1,k+1} &= x_{1,k} + \delta\Big[- w_{3,k}x_{1,k}x_{2,k} - k_2 x_{1,k}x_{3,k} + \\
&\qquad \tau^{-1}\left(w_{1,k} - 2x_{1,k}\right)\Big], \\
x_{2,k+1} &= x_{2,k} + \delta\left[-w_{3,k}x_{1,k}x_{2,k} + \tau^{-1}\left(w_{2,k} - 2x_{2,k}\right)\right], \\
x_{3,k+1} &= x_{3,k} + \delta\left[w_{3,k}x_{1,k}x_{2,k} - k_2 x_{1,k}x_{3,k} - 2\tau^{-1}x_{3,k}\right], \\
x_{4,k+1} &= x_{4,k} + \delta\left[k_2 x_{1,k}x_{3,k} - 2\tau^{-1}x_{4,k}\right].
\end{aligned} \tag{2.58}$$

The parameters $\tau = 20$ (min) and $k_2 = 0.4$ ($\mathrm{M}^{-1}\mathrm{min}^{-1}$) are constant, while $w_{1,k} \in [0.9, 1.1]$ (M), $w_{2,k} \in [0.8, 1.0]$ (M), and $w_{3,k} \in [10, 50]$ ($\mathrm{M}^{-1}\mathrm{min}^{-1}$) are time-varying disturbances. The initial condition is $\mathbf{c}_0 = (0, 0, 0, 0)$.

To apply Method (v), a constraint set G is needed. Since the positive orthant is invariant for the continuous-time system in [27], it can be shown through standard arguments that the solutions of (2.58) are non-negative for $\delta \leq \frac{1}{M}$, where M is as in Corollary 2[29]. However, no other constraints are known to hold for all solutions of (2.58). Thus, we follow the approach in [26] to manufacture valid constraints by defining several redundant state variables:

$$\begin{aligned}
z_{1,k} &= -\frac{1}{3}x_{1,k} - \frac{1}{3}x_{2,k} + \frac{1}{3}x_{3,k}, \\
z_{2,k} &= -\frac{1}{3}x_{1,k} - \frac{1}{3}x_{3,k} + \frac{1}{3}x_{4,k}, \\
z_{3,k} &= -x_{1,k} + 2x_{2,k} + x_{3,k}, \\
z_{4,k} &= x_{1,k} - x_{2,k} + x_{4,k}.
\end{aligned} \tag{2.59}$$

Next, (2.58) is augmented with the redundant equations

$$\begin{aligned}
z_{1,k+1} &= z_{1,k} + \delta\Bigg[w_{3,k}x_{1,k}x_{2,k} - \frac{1}{3}\tau^{-1}\left(w_{1,k}+w_{2,k}\right) \\
&\qquad\qquad - 2\tau^{-1}z_{1,k}\Bigg], \\
z_{2,k+1} &= z_{2,k} + \delta\left[k_2 x_{1,k}x_{3,k} - \frac{1}{3}\tau^{-1}w_{1,k} - 2\tau^{-1}z_{2,k}\right], \\
z_{3,k+1} &= z_{3,k} + \delta\tau^{-1}\left[2\left(w_{2,k}-z_{3,k}\right) - w_{1,k}\right], \\
z_{4,k+1} &= z_{4,k} + \delta\tau^{-1}\left(w_{1,k} - w_{2,k} - 2z_{4,k}\right).
\end{aligned} \tag{2.60}$$

The solutions of this augmented system are, by construction, guaranteed to lie in the set

$$G \equiv \{(k,(\mathbf{x},\mathbf{z}),\mathbf{w}) \in \mathbb{K}\times\mathbb{R}^8\times\mathbb{R}^3 : \mathbf{M}\left[\begin{smallmatrix}\mathbf{x}\\ \mathbf{z}\end{smallmatrix}\right] = \mathbf{b},\ \mathbf{x}\geq\mathbf{0}\}, \tag{2.61}$$

where

$$\mathbf{M} = \begin{bmatrix} -1/3 & -1/3 & 1/3 & 0 & -1 & 0 & 0 & 0 \\ -1/3 & 0 & -1/3 & 1/3 & 0 & -1 & 0 & 0 \\ -1 & 2 & 1 & 0 & 0 & 0 & -1 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0\\0\\0\\0 \end{bmatrix}. \tag{2.62}$$

The specific definition of each $z_{i,k}$ above was chosen such that at least one term cancels out when forming (2.60) from (2.58), resulting in right-hand sides that are bounded more accurately using interval arithmetic.

Since all constraints in G are linear, $\mathcal{I}_G$ is defined by Algorithm 1 and Theorem 4 gives $M_{\mathcal{I}} = 5$. Choosing $\bar{X} = [0,0.13]\times[0,0.13]\times[0,0.45]\times[0,0.5]$ for the original states and $\bar{Z} = [-0.087,0.15]\times[-0.19,0.17]\times[-0.13,0.71]\times[-0.13,0.63]$ for the augmented states, Theorem 2 shows that $M = 13.33$ satisfies (2.11) and (2.28). Thus, by Corollaries 2 and 4, Methods (iv) and (v) are valid with $\delta = 0.015\text{s} \leq \frac{1}{MM_{\mathcal{I}}}$ and $K = 600$.

Figures 2.3 and 2.4 show that the standard interval Method (i) is the most

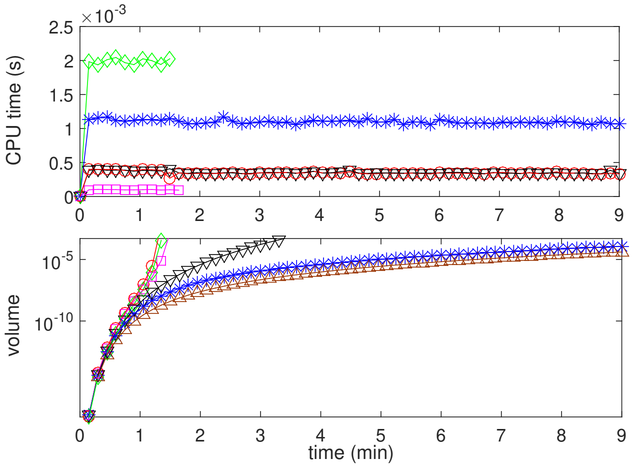

Figure 2.3: Time and enclosure volume for Methods (i)–(v) applied to Example 2.6.3 (□, ○,◇, ▽, ⋆) and the volume of Method (v) bounds intersected with G (△).

computationally efficient but produces weak bounds as expected. The zonotope Methods (ii) and (iii) produce similar bounds, but at higher cost. These methods use linearizations of the dynamics with linearization error bounds computed using an interval Jacobian in Method (ii) and interval Hessians in Method (iii). This strategy appears to be conservative here because the dynamics are highly nonlinear, which leads to large bounds on the linearization error, despite the fact that zonotopes are propagated through the linearized dynamics effectively. Increasing the zonotope order from 10 to 100 did not substantially improve the accuracy of either method, while their costs increased by nearly 10×. The standard DI Method (iv) provides significantly tighter bounds at very low cost. However, these bounds still become weak and diverge after ~ 3 min. In contrast, exploiting G using Method (v) results in bounds that are much tighter than any other method, and remain accurate for most states out to at least 9 min. At the same time, Method (v) requires only $\sim 10^{-3}$ s per time step.

To better understand Method (v), bounds were also computed using only the linear constraints $\mathbf{M}\left[\begin{smallmatrix}\mathbf{x}\\ \mathbf{z}\end{smallmatrix}\right] = \mathbf{b}$ in G and omitting the non-negativity constraints. This resulted in almost no loss of accuracy (the bounds would not be visually distinguishable from the Method (v) bounds in Figure 2.4 and are therefore omitted). In contrast,

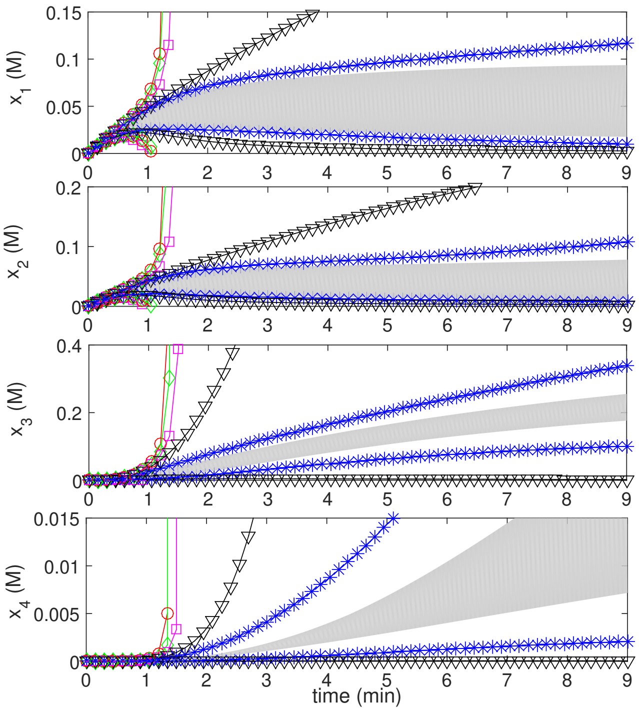

Figure 2.4: Upper and lower bounds on x_1–x_4 in Example 2.6.3 from Methods (i)–(v) (□, ○,◇, ▽, ⋆). Sampled solutions are indicated by the gray shaded region.

eliminating the linear constraints in G and using only non-negativity resulted in a large loss of accuracy, producing nearly the same bounds as Method (iv). Thus, the accuracy of Method (v) is almost entirely due to the manufactured constraints and is not dependent on the fact that the states are non-negative for this system.

To investigate the effects of the initial condition, the comparisons above were repeated with $\mathbf{c}_0 = (0.36, 0.38, 0.36, 0.52)$, which is qualitatively different from $\mathbf{c}_0 = (0, 0, 0, 0)$ in that it is close to the steady-state solution of (2.58). We found that the relative performance of the methods and the qualitative trends in Figures 2.3 and 2.4 did not change with $\mathbf{c}_0$ (results not shown).

2.6.4 Example 4

This example demonstrates the use of nonlinear manufactured constraints using the refinement algorithm developed in Section 2.5.1. We consider the continuous stirred-tank reactor with cooling from [107] in dimensionless form:

$$\begin{aligned}
x_{1,k+1} &= x_{1,k} + \delta\left[w_{1,k} - x_{1,k} - D_a e^{-\frac{\alpha}{x_{2,k}}} x_{1,k}\right], \\
x_{2,k+1} &= x_{2,k} + \delta\Big[w_{2,k} - x_{2,k} - w_{4,k} D_a e^{-\frac{\alpha}{x_{2,k}}} x_{1,k} \\
&\qquad - S\left(x_{2,k} - \beta^{-1} x_{3,k}\right)\Big], \\
x_{3,k+1} &= x_{3,k} + \delta\left[\gamma(w_{3,k} - x_{3,k}) + S_c\left(\beta x_{2,k} - x_{3,k}\right)\right],
\end{aligned} \tag{2.63}$$

where x_1, x_2, and x_3 are the dimensionless concentration, reactor temperature, and cooling water temperature. The time-varying disturbances are the dimensionless inlet concentration, reactor inlet temperature, and inlet cooling water temperature, which are denoted as $w_1 \in [0.8, 1.2]$, $w_2 \in [0.9943, 1.006]$, and $w_3 \in [0.9929, 1.007]$. The dimensionless heat of reaction is treated as a time-invariant uncertainty bounded in $w_4 \in [-0.6097, -0.5858]$. The parameters $D_a = 4.93 \times 10^{11}$, $\alpha = 25$, $\beta = 1.2367$, $\gamma = 1.6096$, $S = 14.3291$, and $S_c = 4.0770$ are constant. The initial condition is $\mathbf{c}_0 = (0.09, 0.9286, 1.127)$.

To apply Method (v), we again use the method in [26] to manufacture a constraint that is useful for refining the state bounds. Specifically, we define the redundant state $z_k = -w_4 x_{1,k} + x_{2,k}$, and augment (2.63) with the redundant equation

$$\begin{aligned}
z_{k+1} = z_k + \delta\Big[&- w_{4,k}(w_{1,k} - x_{1,k}) + w_{2,k} - x_{2,k} \\
&- S\left(x_{2,k} - \beta^{-1} x_{3,k}\right)\Big].
\end{aligned} \tag{2.64}$$

This specific z_k is chosen to cancel the nonlinear reaction term when forming (2.64) from (2.63). The initial condition for the new state satisfies $z_0 \in [0.9813, 0.9834]$. The solutions of the augmented system are now guaranteed to lie in the set

$$G \equiv \{(k, (\mathbf{x}, z), \mathbf{w}) \in \mathbb{K} \times \mathbb{R}^4 \times \mathbb{R}^4 : \mathbf{m}_1(\mathbf{w}) \left(\begin{smallmatrix} \mathbf{x} \\ z \end{smallmatrix} \right) = b_1, \; \mathbf{x} \geq \mathbf{0}\},$$

where $\mathbf{m}_1(\mathbf{w}) = [-w_4 \quad 1 \quad 0 \quad -1]$ and $b_1 = 0$. Since the constraints in G are quadratic, we use the refinement algorithm described in §2.5.1 with the Lipschitz bound $M_{\mathcal{I}} \leq \max(M_{\mathcal{I}Z}, M_{\mathcal{I}V})$ from (2.52). To bound $M_{\mathcal{I}Z}$, we directly apply (2.48) rather than using the simplified bound in (2.49); i.e.,

$$M_{\mathcal{I}Z} \leq \max \left\{ 1, \max_{V \in \mathbb{I}W} \max_{(i,j) \in S(V)} \left\{ \frac{\sum_{l \neq j} |[m_{il}](V)|}{\langle [m_{ij}](V) \rangle} \right\} \right\}. \tag{2.65}$$

Recall that $S(V)$ denotes the set of (i, j) such that $\langle [m_{ij}](V) \rangle \geq \epsilon$. Choosing any $V \in \mathbb{I}W$, the definition of $\mathbf{m}_1$ above gives

$$\begin{aligned} \langle [m_{11}](V) \rangle &= \langle -V_4 \rangle \geq \langle W_4 \rangle = 0.5858, \qquad (2.66) \\ \langle [m_{12}](V) \rangle &= \langle 1 \rangle = 1, \\ \langle [m_{13}](V) \rangle &= \langle 0 \rangle = 0, \\ \langle [m_{14}](V) \rangle &= \langle -1 \rangle = 1. \end{aligned}$$

Choosing $\epsilon = 0.01$, it follows that $S(V) = S(W) = \{(1, 1), (1, 2), (1, 4)\}$ for all $V \in \mathbb{I}W$. Therefore, $S(V)$ can be replaced by $S(W)$ in (2.65), which permits the order of the maximizations to be reversed, so that

$$\begin{aligned} M_{\mathcal{I}Z} &\leq \max \left\{ 1, \max_{(i,j) \in S(W)} \max_{V \in \mathbb{I}W} \left\{ \frac{\sum_{l \neq j} |[m_{il}](V)|}{\langle [m_{ij}](V) \rangle} \right\} \right\}, \qquad (2.67) \\ &\leq \max \left\{ 1, \max_{(i,j) \in S(W)} \left\{ \frac{\sum_{l \neq j} |[m_{il}](W)|}{\langle [m_{ij}](W) \rangle} \right\} \right\}, \\ &= 3.41, \end{aligned}$$

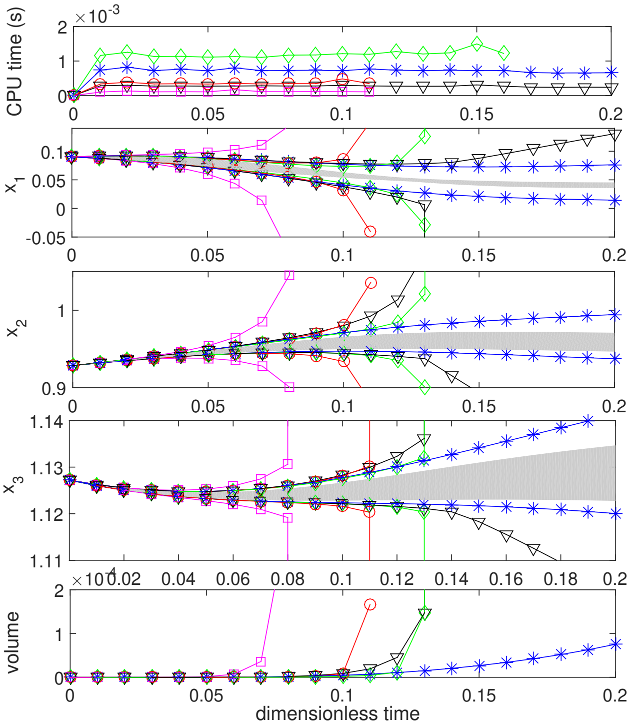

Figure 2.5: Time, upper and lower bounds on x_1–x_3, and enclosure volume for Methods (i)–(v) applied to Example 2.6.4 (□, ○,◇, ▽, ⋆). Sampled solutions are indicated by the gray shaded region.

where the second inequality follows from inclusion monotonicity of $[\mathbf{m}_1]$ [102].

The simple bound (2.51) gives $M_{\mathcal{I}V} \leq \epsilon^{-1}\|\boldsymbol{\gamma}_1\|_1 = 200$. A sharper bound can potentially be obtained using (2.50), which depends on $\bar{X}$ and simplifies to $M_{\mathcal{I}V} \leq 2(\bar{x}_1^L)^{-1}$ assuming $\bar{X}_1$ is non-negative. Simulation data show that $\bar{X} = [0.01, 0.1] \times [0.92, 1] \times [1, 1.17] \times [0.92, 1.1]$ is a reasonable choice, which leads again to $M_{\mathcal{I}V} \leq 200$. To avoid this large constant, we chose to omit lines 13–18 in the modified algorithm in §2.5.1, as described in Remark 4, which results in the final bound $M_{\mathcal{I}} \leq M_{\mathcal{I}Z} \leq 3.41$. Remark 2 shows that $M = 272.78$ satisfies (2.11) and (2.28). Thus, by Corollaries 2 and 4, Methods (iv) and (v) are valid with $\delta = 0.001\text{s} \leq \frac{1}{MM_{\mathcal{I}}}$ and $K = 200$.

Figure 2.5 shows the resulting state bounds, enclosure volume, and the CPU time for Methods (i)–(v). The interval Method (i) again provides the weakest bounds. The zonotope Methods (ii) and (iii) both provide improved bounds, with Method (iii) being tighter but also more expensive. However, neither method remains accurate beyond a dimensionless time of 0.1. As in Example 2.6.3, these methods appear to be conservative here because the dynamics are highly nonlinear, leading to large bounds on the linearization errors. Increasing the zonotope order from 10 to 100 did not substantially improve the accuracy of either method, while their costs increased by nearly 10×. The standard DI Method (iv) produces bounds that are comparable to those of Method (iii), but is significantly more computationally efficient. Finally, Method (v) is again the most effective, producing tight bounds well beyond 0.1 for a cost intermediate between that of Method (iv) and Method (iii).

We also tested Method (v) with the non-negativity constraints removed from G, which resulted in almost no loss of accuracy, and with the manufactured constraint removed from G but non-negativity retained, which resulted in a large loss of accuracy, producing bounds only slightly better than Method (iv) (not shown). As in Example 2.6.3, we conclude that the accuracy of Method (v) is almost entirely due to the manufactured constraint and is not dependent on non-negativity of the states.

2.6.5 Example 5

This example shows the application and limitations of Methods (iv) and (v) for general (non-Euler) discrete-time systems. This system was used in [23] to test Method (ii):

$$\mathbf{x}_{k+1} = \begin{bmatrix} 0 & -0.5 \\ 1 & 1+0.3v_k \end{bmatrix} \mathbf{x}_k + 0.02 \begin{bmatrix} -6 \\ 1 \end{bmatrix} \mathbf{w}_k, \tag{2.68}$$

where $\mathbf{w}_k \in [-1,1]\times[-1,1]$, $v_k \in [-1,1]$, and $\mathbf{c}_0 \in [-3,3]\times[-3,3]$. Since $1+0.3v_k > 0$ for every $v_k \in [-1,1]$, the monotonicity condition (2.8) required by Corollary 1 holds, and hence the standard DI Method (iv) is valid. Unfortunately, Method (v) is not valid for any choice of G and $\mathcal{I}_G$ because the monotonicity condition (2.25) required by Corollary 3 fails for any $M_{\mathcal{I}} > 0$. This is because $\frac{\partial h_1}{\partial x_1} = 0$, while $|\frac{\partial h_1}{\partial x_2}| = 0.5 > 0$.

Numerical results (not shown) show that the standard interval Method (i) produces rapidly diverging bounds. The zonotope Methods (ii) and (iii) are much more accurate but have significant overestimation that grows roughly linearly with time. Method (iv) yields exactly the same bounds as Method (i), and is therefore not competitive with the zonotope methods in this case.

Due to their close relationship to effective continuous-time DI methods, we expect Methods (iv) and (v) to perform well for discrete-time systems that accurately approximate a continuous-time system (by Euler discretization or otherwise). Any such system must have relatively large positive values of $\frac{\partial h_i}{\partial x_i}$ for all i. In contrast, (2.68) has $\frac{\partial h_1}{\partial x_1} = 0$, and this is precisely why it is impossible to apply Method (v).

The equivalence of Methods (i) and (iv) is actually an indication that this problem is (relatively) good for Method (i) rather than bad for Method (iv). Specifically, this occurs because each x_i only appears once in the corresponding h_i (i.e., there is no interval dependency problem [102]). In this case, the simple interval extension of h_i over X_k used in Method (i) gives the same result as the interval extensions of h_i over the faces $\beta_i^{L/U}(X_k)$ used in (2.5). Similarly, the strength of the zonotope Methods (ii) and (iii) relative to Method (iv) indicates that these methods are particularly well suited for this problem. This is because, in contrast to Examples 2.6.3 and 2.6.4, the dynamics are nearly linear, leading to small linearization errors (i.e., almost all elements in the Jacobian and Hessian matrices used in Methods (ii) and (iii) are constant, so their interval extensions are accurate).

2.6.6 Example 6

The Lotka-Volterra model was used to test Method (iii) in [22]. After forward Euler discretization, the model is

$$\begin{aligned} x_{k+1,1} &= x_{k,1} + \delta(a x_{k,1} - b x_{k,1} x_{k,2}), \\ x_{k+1,2} &= x_{k,2} + \delta(-c x_{k,2} + d x_{k,1} x_{k,2}), \end{aligned} \tag{2.69}$$

where $a = c = 1$, $b = 0.01$, $d = 0.02$, and $\mathbf{c}_0 \in [49, 50] \times [49, 50] \times [147, 150]$. To implement Method (v), we again manufacture a constraint as in [26] by defining

$$z_k = 100 d x_{k,1} + 100 b x_{k,2}$$

and augmenting (2.69) with the redundant equation $z_{k+1} = z_k + \delta(100 da x_{k,1} - 100 bc x_{k,2})$. Thus, we define

$$G \equiv \{(k, (\mathbf{x}, z)) \in \mathbb{K} \times \mathbb{R}^3 : \mathbf{m}_1^{\mathrm{T}} \left(\begin{smallmatrix} \mathbf{x} \\ z \end{smallmatrix}\right) = b_1\}, \tag{2.70}$$

where $\mathbf{m}_1^{\mathrm{T}} = [2 \quad 1 \quad -1]$ and $b_1 = 0$.

Theorem 4 in [106] gives the Lipschitz constant $M_{\mathcal{I}} = 3$. Moreover, choosing $\bar{X} = [0.01, 250] \times [20, 350] \times [39.6, 962]$ for the augmented system, Remark 2 shows that $M = 11$ satisfies (2.11) and (2.28). Thus, by Corollaries 2 and 4, Methods (iv) and (v) are valid with $\delta = 0.01 \leq \frac{1}{M M_{\mathcal{I}}}$ and $K = 550$.

Figure 2.6 shows that the standard interval Method (i) is computationally efficient but very conservative. In contrast, both zonotope methods produce very tight bounds, with Method (iii) slightly tighter but also less computationally efficient than Method (ii). Unlike Examples 2.6.3 and 2.6.4, these methods perform well because the Jacobain matrix used in Method (ii) is fairly simple, while the Hessians used in Method (iii)

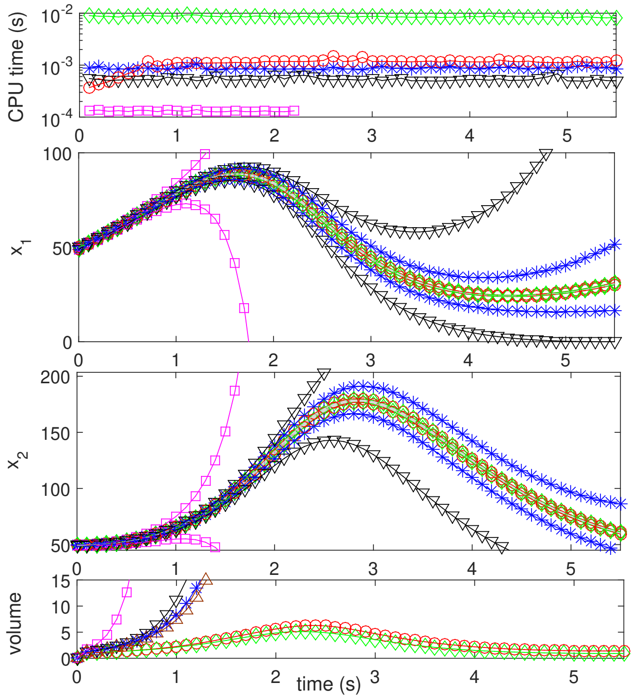

Figure 2.6: Time, upper and lower bounds on x_1–x_2, and enclosure volume for Methods (i)–(v) applied to Example 2.6.6 (□, ○,◇, ▽, ⋆) and the volume of Method (v) bounds intersected with G (△). Sampled solutions are indicated by the gray shaded region.

are constant, leading to tight bounds on the linearization error in both methods. Relative to the zonotope methods, the standard DI Method (iv) is significantly more conservative, while Method (v) is competitive but still notably more conservative.

To provide another illustration of the use of nonlinear constraints, we now consider the case with uncertain b and d. To obtain a reasonable M satisfying (2.28), it proves useful to scale these parameters. Thus, we define $\mathbf{w} = (w_1, w_2) = (100b, 100d)$ and let $\mathbf{w} \in W \equiv [0.98, 1.02] \times [1.98, 2.02]$. Writing the right-hand sides of (2.69) as functions of $\mathbf{w}$, Remark 2 shows that $M = 19.87$ satisfies (2.11) and (2.28).

With uncertain a and b, the manufactured constraint above becomes quadratic. Specifically, we define

$$G \equiv \{(k, (\mathbf{x}, z), \mathbf{w}) \in \mathbb{K} \times \mathbb{R}^3 \times \mathbb{R}^2 : \mathbf{m}_1^{\mathrm{T}}(\mathbf{w}) \left(\begin{smallmatrix}\mathbf{x}\\ z\end{smallmatrix}\right) = 0\}, \tag{2.71}$$

where $\mathbf{m}_1^{\mathrm{T}} = [w_2 \quad w_1 \quad -1]$. We define $\mathcal{I}_G$ as in Algorithm 2 with the modifications for quadratic constraints in Section 2.5.1. To compute $M_{\mathcal{I}Z}$, we follow the same procedure outlined in Example 2.6.4. For any, $V \in \mathbb{I}W$, the definition of $\mathbf{m}_1$ gives

$$\begin{aligned}\langle [m_{11}](V) \rangle &= \langle V_2 \rangle \geq \langle W_2 \rangle = 1.98, \\ \langle [m_{12}](V) \rangle &= \langle V_1 \rangle \geq \langle W_1 \rangle = 0.98, \\ \langle [m_{13}](V) \rangle &= \langle -1 \rangle = 1.\end{aligned} \tag{2.72}$$

Choosing $\epsilon = 0.01$, it follows that $S(V) = \{(1,1),(1,2),(1,3)\}$ for all $V \in \mathbb{I}W$. Thus, by (2.48),

$$M_{\mathcal{I}Z} \leq \max \left\{ 1, \max_{(i,j) \in S(W)} \left\{ \frac{\sum_{l \neq j} |[m_{il}](W)|}{\langle [m_{ij}](W) \rangle} \right\} \right\} = 3.1.$$

As in Example 2.6.4, (2.50) provided a large bound for $M_{\mathcal{I}V}$. To avoid this, we chose to omit lines 13–18 in the modified algorithm in §2.5.1, as described in Remark 4, which results in the final bound $M_{\mathcal{I}} \leq M_{\mathcal{I}Z} \leq 3.1$. By Corollaries 2 and 4, Methods (iv) and (v) are valid with $\delta = 0.01 \leq \frac{1}{M M_{\mathcal{I}}}$ and $K = 550$.

Figure 2.7 shows that the zonotope methods are again the most effective. However, Method (ii) is significantly more conservative in this case and diverges after ~ 5 s. The DI methods (iv) and (v) are both more conservative than Method (iii). However, Method (v) is significantly tighter than Method (iv) and diverges more slowly, eventually becoming tighter than Method (ii) as well. Although Method (iii) remains the best for this example, the bounds are not as accurate as when a and d were certain because now the Hessian matrices contain uncertain elements.

Fig. 2.8 shows the results of Methods (i)–(v) with the larger uncertainty set $W = [0.9, 1.05] \times [1.9, 2.05]$. None of the methods is able to produce accurate bounds in this case. Notably, however, the zonotope Methods (ii)–(iii) are disproportionately

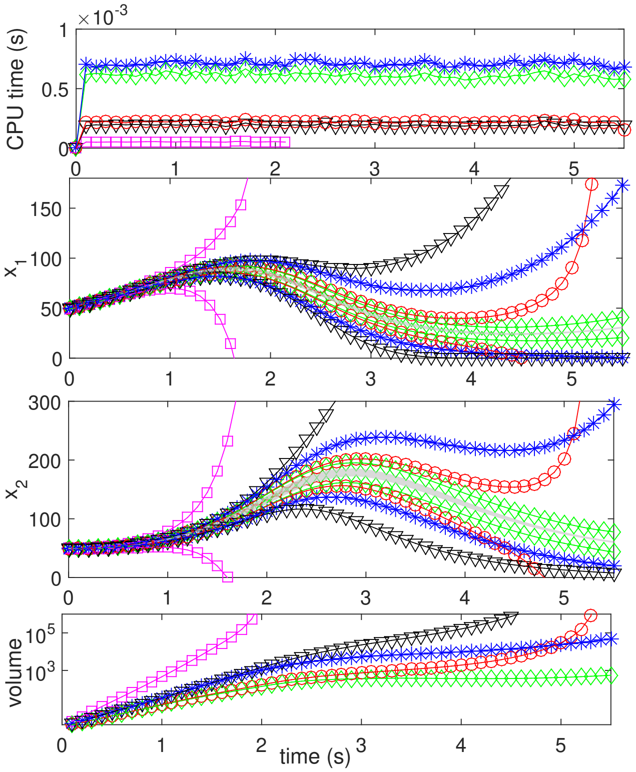

Figure 2.7: Time, upper and lower bounds on x_1–x_2, and enclosure volume for Methods (i)–(v) applied to Example 2.6.6 with $W = [0.98, 1.02] \times [1.98, 2.02]$ (□, ○,◇, ▽, ⋆). Sampled solutions are indicated by the gray shaded region.

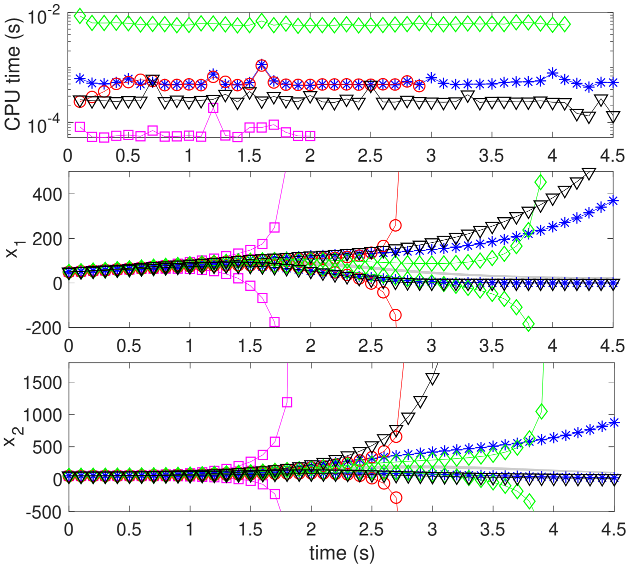

Figure 2.8: Time and upper and lower bounds on x_1–x_2 for Methods (i)–(v) applied to Example 2.6.6 with $W = [0.9, 1.05] \times [1.9, 2.05]$ (□, ○,◇, ▽, ⋆). Sampled solutions are indicated by the gray shaded region.

affected by the increased uncertainty, which adversely affect the linearization error bounds used in these methods. Both methods diverge rapidly prior to 4 s. In contrast, the DI Methods (iv)–(v) diverge much more slowly, with Method (iv) providing tighter bounds than Method (ii) after 3 s and Method (v) outperforming all other methods after 4 s.

2.7 Conclusion

Effective reachable set bounding methods for continuous-time systems based on differential inequalities (DI) have been extended to discrete-time systems under sufficient monotonicity conditions. For Euler discretized systems, these conditions are always satisfied when using a step size below a computable upper bound. This bound depends on Lipschitz constants for the dynamics and for the refinement operator used to exploit the (possibly redundant) system constraints, if any. Two new refinement algorithms were proposed for exploiting nonlinear constraints in a way that effectively balances accuracy with the need to achieve a small Lipschitz constant. Examples 2.6.3 and 2.6.4 show that the standard DI method can provide significant gains in accuracy at lower cost when compared with existing bounding approaches based zonotopes, while advanced DI methods using refinements based on redundant model equations provide much more accurate bounds at similar cost. However, Examples 2.6.5–2.6.6 show that zonotopic approaches are still more effective for some problems. Our results suggest that zonotopic methods are more effective when the interval Jacobian or Hessian matrices used for bounding the linearization errors in these methods have few uncertain elements. In contrast, the DI approaches appear to be more effective for highly nonlinear system with large uncertainties, particularly when nonlinear or uncertain terms can be canceled through the introduction of appropriate new variables and manufactured constraints.

CHAPTER 3

ACCURATE SET-BASED STATE ESTIMATION FOR NONLINEAR DISCRETE-TIME SYSTEMS USING DIFFERENTIAL INEQUALITIES WITH MODEL REDUNDANCY

3.1 INTRODUCTION

This chapter presents a new set-based state estimation algorithm for nonlinear discrete-time systems subject to bounded disturbances and measurement errors. In contrast to conventional state estimation, which aims to compute a single best estimate for the current state, set-based state estimation aims to compute a set that rigorously encloses all states consistent with the given model and the observed outputs up to the present time. Set-based state estimation is central to a variety of algorithms for guaranteed fault diagnosis and robust control [8, 32, 7, 30, 31, 33].

Set-based state estimation is typically done recursively in two steps. Given a set of consistent states $\hat{X}_{k|k}$ at time k, the *prediction* step computes an enclosure $\hat{X}_{k+1|k}$ of the states reachable at $k+1$ from $\hat{X}_{k|k}$ by the given dynamics. In the *correction* step, $\hat{X}_{k+1|k}$ is refined to produce $\hat{X}_{k+1|k+1}$ by eliminating regions of $\hat{X}_{k+1|k}$ that are inconsistent with the observed output at $k+1$. Algorithmically, prediction requires bounding the image of $\hat{X}_{k|k}$ under a nonlinear function, while correction requires bounding the intersection of two sets.

Starting with the seminal papers [36, 108], a wide variety of methods have been developed for set-based state estimation using intervals, ellipsoids, parallelotopes, polytopes, zonotopes, and constrained zonotopes [15, 35, 13, 14, 2, 4]. However, methods for both prediction and correction often suffer from large overestimation errors or high computational costs, particularly for nonlinear systems with large uncertainties. In [15, 16], efficient interval computations are used for both steps. However, achieving accurate results often requires extensive partitioning, which can lead to very high costs. Several alternative approaches [109, 22, 23, 37] use linearized dynamics with rigorous error bounds in order to apply efficient prediction methods for linear systems using ellipsoids and zonotopes [36, 2]. However, these methods are vulnerable to large linearization errors [106]. Moreover, the correction step requires bounding the intersection of an ellipsoid or a zonotope with a measurement set, which remains a source of significant overestimation and computational complexity (see [36, 22] for simple heuristics and [35, 23, 37] for more accurate online optimization or enumeration procedures). A promising new zonotope-based correction step for systems with linear output equations is presented in [4]. Finally, the method in [19] aims to achieve more accurate predictions by solving difference of convex functions (DC) programs online, but requires an effective DC representation of the dynamics.

This chapter presents a new set-based state estimation algorithm with an improved prediction step based on the theory of differential inequalities (DI). Methods based on DI have been extensively developed for bounding the reachable sets of nonlinear systems in continuous-time [24, 26, 99, 100, 27] (note that reachability bounding differs from set-based state estimation in that measurements are not considered). The most basic DI method uses only simple interval computations, and therefore provides bounds that are very efficient but often very weak. Although some DI approaches have addressed this using more complex sets [99, 27], our interest here is in methods that achieve tighter bounds at low cost using *model redundancy*. These methods use efficient bounds-tightening techniques based on redundant equations that are implied by the given dynamics, but are not necessarily preserved by the computed bounds due to overestimation. In many applications, such redundant equations are readily available in the form conservation laws, physical bounds on certain states, etc. [24]. More generally, any system of interest can be embedded in a higher-dimensional system that obeys such relationships by design, as described in detail in [26] (see also §3.5). Extensive numerical experiments in [24, 26] show that using model redundancy enables fast interval-based DI methods to produce very sharp reachability bounds for a variety of challenging problems. Recently, the authors extended this approach to address reachability problems in discrete-time with similar results [106]. Notably, this approach has so far only been shown to be valid for forward-Euler-discretized systems satisfying a step size bound. However, many discrete-time systems are formed in this way in practice, and the step size bound is not very restrictive [106].

In this chapter, we develop a new set-based state estimation algorithm by adapting the DI-based reachability method in [106] to provide accurate prediction sets using only fast interval computations. The prediction step of our algorithm is not quite a direct application of the method in [106]. Instead, we show that output measurements can be used to modify the prediction step in a simple but nontrivial way, leading to significantly tighter prediction bounds. This method is described in detail in §3.4, following the formal problem statement and background information in §3.2–3.3. In §3.5, we show that this method produces state estimates with significantly higher accuracy and efficiency than state-of-the-art zonotopic methods for a challenging nonlinear chemical reactor model.

3.2 Problem Statement

We consider nonlinear discrete-time systems in the following forward-Euler-discretized form with step size $h \in \mathbb{R}_+$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\mathbf{f}\left(\mathbf{x}_k, \mathbf{w}_k\right), \tag{3.1}$$

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k, \mathbf{v}_k). \tag{3.2}$$

Above, $\mathbf{x}_k \in \mathbb{R}^{n_x}$ is the state, $\mathbf{y}_k \in \mathbb{R}^{n_y}$ is the output, $\mathbf{w}_k \in \mathbb{R}^{n_w}$ is the disturbance, $\mathbf{v}_k \in \mathbb{R}^{n_v}$ is the measurement error, $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$ is locally Lipschitz continuous, $\mathbf{g} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_y}$, and $k \in \mathbb{K} \equiv \{0, \ldots, K\}$ with horizon length K. The initial conditions, disturbances, and measurement errors are assumed to lie in known compact intervals,

$$(\mathbf{x}_0, \mathbf{w}_k, \mathbf{v}_k) \in C_0 \times W \times V, \quad \forall k \in \mathbb{K}. \tag{3.3}$$

Our objective is to compute accurate enclosures of $X_{k|k}(\mathbf{y}_{0:K})$ and $X_{k+1|k}(\mathbf{y}_{0:K})$ defined below, which contain all states at k and $k+1$, respectively, that are consistent with (3.1)–(3.3) and an observed output sequence $\mathbf{y}_{0:K} = (\mathbf{y}_0, \ldots, \mathbf{y}_K)$ up to k. For any $\mathbf{y} \in \mathbb{R}^{n_y}$, define the measurement set

$$X^m(\mathbf{y}) \equiv \{\mathbf{x} \in \mathbb{R}^{n_x} : \mathbf{y} = \mathbf{g}\left(\mathbf{x}, \mathbf{v}\right), \mathbf{v} \in V\}. \tag{3.4}$$

Then, $X_{k|k}(\mathbf{y}_{0:K})$ and $X_{k+1|k}(\mathbf{y}_{0:K})$ are defined precisely for all $k \in \mathbb{K}$ by the following recursion:

$$X_{0|-1}(\mathbf{y}_{0:K}) \equiv C_0, \tag{3.5}$$

$$X_{k|k}(\mathbf{y}_{0:K}) \equiv X_{k|k-1}(\mathbf{y}_{0:K}) \cap X^m(\mathbf{y}_k), \tag{3.6}$$

$$X_{k+1|k}(\mathbf{y}_{0:K}) \equiv \{\mathbf{x} + h\mathbf{f}(\mathbf{x}, \mathbf{w}) : (\mathbf{x}, \mathbf{w}) \in X_{k|k}(\mathbf{y}_{0:K}) \times W\}. \tag{3.7}$$

We assume throughout the chapter that a crude set G is known *a priori* to contain all solutions of (3.1), irrespective of the observed output $\mathbf{y}_{0:K}$. Let $\mathbf{x}_k(\mathbf{c}_0, \mathbf{w}_{0:K})$ denote the solution of (3.1) at k with initial condition $\mathbf{c}_0 \in C_0$ and disturbances $\mathbf{w}_{0:K} \in W_{0:K} \equiv W \times \cdots \times W$.

Assumption 4. A set $G \subset \mathbb{R}^{n_x}$ is known such that

$$\mathbf{x}_k(\mathbf{c}_0, \mathbf{w}_{0:K}) \in G \tag{3.8}$$

for all $(\mathbf{c}_0, \mathbf{w}_{0:K}) \in C_0 \times W_{0:K}$ and all $k \in \{0, \dots, K+1\}$.

Assumption 4 is not restrictive since choosing $G = \mathbb{R}^{n_x}$ is always valid. However, in many applications, a nontrivial set G can be defined as the set of points satisfying relevant conservation laws, physical bounds, etc. These constraints are redundant in the sense that they are implied by the dynamics. Nevertheless, they are often violated by the conservative set-based calculations used in reachability analysis and set-based state estimation [24]. As outlined in §3.1, the new state estimation algorithm presented here is based on the reachable set bounding method in [106], which is able to use nontrivial G sets to achieve much tighter enclosures in many cases. Moreover, as shown originally in [26], a nontrivial set G can be obtained for nearly any system of interest by embedding it in a higher-dimensional system whose states satisfy a set of equality constraints by definition, and this often leads to much tighter reachability bounds. We show an example of this simple construction §3.5.

3.3 Discrete-Time Differential Inequalities

This section briefly reviews the main result of [106] for bounding the reachable set of (3.1), which will be used for the prediction step of the new set-based state estimator presented in §3.4. The reachability problem addressed in [106] is to compute tight interval bounds X_k such that

$$\mathbf{x}_k(\mathbf{c}_0, \mathbf{w}_{0:K}) \in X_k \tag{3.9}$$

for all $(\mathbf{c}_0, \mathbf{w}_{0:K}) \in C_0 \times W_{0:K}$ and all $k \in \{0, \dots, K+1\}$.

We begin by introducing some basic notation. For $\mathbf{z}^L, \mathbf{z}^U \in \mathbb{R}^n$, let $Z = [\mathbf{z}^L, \mathbf{z}^U]$ denote the compact n-dimensional interval $\{\mathbf{z} \in \mathbb{R}^n : \mathbf{z}^L \leq \mathbf{z} \leq \mathbf{z}^U\}$, and denote the set of all such intervals by $\mathbb{IR}^n$. An interval-valued function $H : \mathbb{IR}^n \to \mathbb{IR}^m$ is called an *inclusion function* for $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^m$ if, for every $Z \in \mathbb{IR}^n$, we have $\mathbf{h}(Z) \equiv \{\mathbf{h}(\mathbf{z}) : \mathbf{z} \in Z\} \subset H(Z)$. We assume throughout that an inclusion function $F : \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_w} \to \mathbb{IR}^{n_x}$ is available for $\mathbf{f}$ in (3.1), e.g. by interval arithmetic, and we denote its elements by $F_i(X, W) = [f_i^L(X, W), f_i^U(X, W)]$.

To state the main reachability result in [106], two kinds of interval operators must be defined. The first are the *flattening* or *face selection* operators $\beta_i^L, \beta_i^U : \mathbb{IR}^n \to \mathbb{IR}^n$ defined for every $i \in \{1, \ldots, n\}$ by

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

The second is a generic *refinement operator* that bounds the intersection of an interval Z with an arbitrary set A, and is required to satisfy a Lipschitz continuity property.

Definition 4. Given any set $A \subset \mathbb{R}^n$, let $\mathcal{I}[\cdot, A] : \mathbb{IR}^n \to \mathbb{IR}^n$ satisfy the following conditions:

1. $(Z \cap A) \subset \mathcal{I}[Z, A] \subset Z$, $\forall Z \in \mathbb{IR}^n$.

2. $\mathcal{I}[\cdot, A]$ is Lipschitz continuous with respect to the Hausdorff metric d_H; i.e., $\exists M_A \in \mathbb{R}_+$ such that

$$d_H(\mathcal{I}[Z, A], \mathcal{I}[\hat{Z}, A]) \leq M_A d_H(Z, \hat{Z}), \quad \forall Z, \hat{Z} \in \mathbb{IR}^n.$$

A specific refinement algorithm $\mathcal{I}[\cdot, A]$ satisfying Definition 4 for polyhedral sets A is available in [106] and is generalized here in §3.4.

With some technical caveats, the central result of [106] is that valid reachability bounds $X_k = [\mathbf{x}_k^L, \mathbf{x}_k^U]$ satisfying (3.9) are given by the solutions of the following dynamic system with $X_0 = C_0$:

$$x_{k+1,i}^L = x_{k,i}^L + hf_i^L(\mathcal{I}[\beta_i^L(X_k), G], W), \tag{3.10}$$

$$x_{k+1,i}^U = x_{k,i}^U + hf_i^U(\mathcal{I}[\beta_i^U(X_k), G], W). \tag{3.11}$$

To understand the key ideas, consider first the case where $G = \mathbb{R}^{n_x}$ and $\mathcal{I}[Z, G] = Z$ for all $Z \in \mathbb{IR}^{n_x}$, and assume inductively that, for some $k \in \mathbb{K}$, we have $\mathbf{x}_k(\mathbf{c}_0, \mathbf{w}_{0:K}) \in X_k$ for all $(\mathbf{c}_0, \mathbf{w}_{0:K}) \in C_0 \times W_{0:K}$. Then, this result states that, e.g., a valid lower bound on $x_{k+1,i}(\mathbf{c}_0, \mathbf{w}_{0:K})$ is obtained by bounding the range of f_i over the interval $\beta_i^L(X_k) \times W$. The idea of bounding f_i only over the i^{th} lower face of X_k rather than over all of X_k is central to reachable set bounding methods for continuous-time systems based on differential inequalities (DI), and is related to the simple observation that a continuous-time trajectory cannot leave a continuous, time-varying interval enclosure without being incident on its boundary at some point in time. For an arbitrary sequence of intervals X_k in discrete-time, the analogous claim that $x_{k,i}(\mathbf{c}_0, \mathbf{w}_{0:K})$ must coincide with $x_{k,i}^L$ in order for $x_{k+1,i}(\mathbf{c}_0, \mathbf{w}_{0:K})$ to lie below $x_{k+1,i}^L$ is clearly not true. Nevertheless, the main result of [106] shows that the use of $\beta_i^{L/U}$ in (3.10)–(3.11) still produces valid bounds provided that the step size h is below an upper bound. Moreover, this upper bound is easily computable and was shown to be reasonable in practice.

When the *a priori* enclosure G is nontrivial, (3.10)–(3.11) states that the faces $\beta_i^{L/U}(X_k)$ can be further refined by eliminating regions that lie outside of G before bounding the range of each f_i. Note that G is permitted to refine each face of X_k independently, rather than refining X_k first and selecting faces second, as in $\beta_i^{L/U}(\mathcal{I}[X_k, G])$. Unless G is an interval, it is generally true that $\mathcal{I}[\beta_i^{L/U}(X_k), G] \subset$

$\beta_i^{L/U}(\mathcal{I}[X_k, G])$, so this often has a very significant impact on the accuracy of the resulting reachability bounds.

The use of the flattening operators $\beta_i^{L/U}$ and the model redundancy G to reduce overestimation in (3.10)–(3.11) is unique among discrete-time bounding algorithms, and was shown to be very effective relative to state-of-the-art zonotopic methods in [106]. However, these techniques have not previously been applied in the context of state estimation.

3.4 Set-Based State Estimation using Discrete-Time Differential Inequalities

This section presents a new set-based state estimation algorithm whose prediction step is based on the discrete-time differential inequalities approach outlined in the previous section. Let $\mathbf{y}_{0:K}$ be an observed output sequence for (3.1)–(3.2) and recall the definitions of $X_{k|k}(\mathbf{y}_{0:K})$ and $X_{k+1|k}(\mathbf{y}_{0:K})$ from §3.2. Furthermore, recall the measurement set $X^m(\mathbf{y})$ from §3.2 and, for every $X \in \mathbb{IR}^{n_x}$ and $\mathbf{y} \in \mathbb{R}^{n_y}$, define the shorthand

$$\Omega(X, \mathbf{y}) \equiv \mathcal{I}[X, X^m(\mathbf{y}) \cap G], \tag{3.12}$$

$$\Omega_i^L(X, \mathbf{y}) \equiv \mathcal{I}[\beta_i^L(X), X^m(\mathbf{y}) \cap G], \tag{3.13}$$

$$\Omega_i^U(X, \mathbf{y}) \equiv \mathcal{I}[\beta_i^U(X), X^m(\mathbf{y}) \cap G]. \tag{3.14}$$

In Theorem 6 below, we will show that intervals $\hat{X}_{k|k} = [\hat{\mathbf{x}}_{k|k}^L, \hat{\mathbf{x}}_{k|k}^U]$ and $\hat{X}_{k+1|k} = [\hat{\mathbf{x}}_{k+1|k}^L, \hat{\mathbf{x}}_{k+1|k}^U]$ that enclose $X_{k|k}(\mathbf{y}_{0:K})$ and $X_{k+1|k}(\mathbf{y}_{0:K})$, respectively, are given by the following recursive algorithm:

$$\hat{X}_{0|-1} = C_0, \tag{3.15}$$

$$\hat{X}_{k|k} = \Omega(\hat{X}_{k|k-1}, \mathbf{y}_k), \tag{3.16}$$

$$\hat{x}^L_{k+1|k,i} = \hat{x}^L_{k|k,i} + hf^L_i(\Omega^L_i(\hat{X}_{k|k}, \mathbf{y}_k), W), \tag{3.17}$$

$$\hat{x}^U_{k+1|k,i} = \hat{x}^U_{k|k,i} + hf^U_i(\Omega^U_i(\hat{X}_{k|k}, \mathbf{y}_k), W). \tag{3.18}$$

Note that applying the reachability method described in the previous section directly to the prediction step here would have resulted in predictions (3.17)–(3.18) using the alternative definitions $\Omega^{L/U}_i(X, \mathbf{y}) = \mathcal{I}[\beta^{L/U}_i(X), G]$. In this case, the measurement $\mathbf{y}_k$ would only be used to refine $\hat{X}_{k|k-1}$ in correction step (3.16). However, as written, the estimator (3.15)–(3.18) also uses $\mathbf{y}_k$ to refine the individual faces of $\hat{X}_{k|k}$ before bounding the ranges of the functions f_i. We show in §3.5 that this can lead to significantly tighter enclosures when the output $\mathbf{y}_k$ is not simply a subset of the states $\mathbf{x}_k$.

Theorem 6. *Choose any* $\mathbf{y}_{0:K} \in \mathbb{R}^{(K+1)n_y}$*, let* G *satisfy Assumption 4, let* $\mathcal{I}[\cdot, X^m(\mathbf{y}_k) \cap G]$ *satisfy Definition 4 for every* $k \in \mathbb{K}$ *with the same Lipschitz constant* $M_G \in \mathbb{R}_+$*, and let* $\hat{X}_{k|k}$ *and* $\hat{X}_{k+1|k}$ *be defined for all* $k \in \mathbb{K}$ *by* (3.15)–(3.18) *with the definitions* (3.12)–(3.14)*. Furthermore, choose any compact set* $\bar{X} \subset \mathbb{R}^{n_x}$ *containing* C_0 *and let* $M \in \mathbb{R}_+$ *satisfy*

$$\|\mathbf{f}(\mathbf{z}, \mathbf{w}) - \mathbf{f}(\hat{\mathbf{z}}, \mathbf{w})\|_\infty \leq M\|\mathbf{z} - \hat{\mathbf{z}}\|_\infty, \tag{3.19}$$

for all $(\mathbf{z}, \hat{\mathbf{z}}, \mathbf{w}) \in \bar{X} \times \bar{X} \times W$*. Let* K^* *be the largest* $k \in \mathbb{K}$ *such that* $\hat{X}_{k|k} \subset \bar{X}$*. If* $h \in (0, \frac{1}{MM_G}]$*, then*

$$X_{k|k}(\mathbf{y}_{0:K}) \subset \hat{X}_{k|k} \quad \textit{and} \quad X_{k+1|k}(\mathbf{y}_{0:K}) \subset \hat{X}_{k+1|k}, \tag{3.20}$$

for all $k \in \mathbb{K}^* \equiv \{0, \dots, K^*\}$.

Remark 5. If $\mathbf{f}$ is globally Lipschitz, then (3.19) holds without restricting $\mathbf{z}$ and $\hat{\mathbf{z}}$ to a compact set $\bar{X}$. Thus, there is no need to specify $\bar{X}$ and we can set $K^* = K$. If $\mathbf{f}$ is only locally Lipschitz, then M is only guaranteed to exist for $\mathbf{z}, \hat{\mathbf{z}} \in \bar{X}$, and M depends on the choice of $\bar{X}$. In applying Theorem 6, we simply choose $\bar{X}$ as a reasonably large interval based on physical insight, and then check $\hat{X}_{k|k} \subset \bar{X}$ at every time step. When $\bar{X}$ is an interval and $\mathbf{f}$ is continuously differentiable, Theorem 2 in [106] provides a simple means to compute M, which we use for all examples in §3.5. Computing M_G is discussed following the proof.

Proof. To set up an inductive argument, note that $X_{0|-1}(\mathbf{y}_{0:K}) \subset \hat{X}_{0|-1}$ since both sets equal C_0 by definition. Choose any $k \in \mathbb{K}^*$ and assume that $X_{k|k-1}(\mathbf{y}_{0:K}) \subset \hat{X}_{k|k-1}$. We will show that this implies $X_{k|k}(\mathbf{y}_{0:K}) \subset \hat{X}_{k|k}$ and $X_{k+1|k}(\mathbf{y}_{0:K}) \subset \hat{X}_{k+1|k}$. The results then follows by induction.

We first show that $X_{k|k}(\mathbf{y}_{0:K}) \subset \hat{X}_{k|k}$. By definition, $X_{k|k}(\mathbf{y}_{0:K}) = X_{k|k-1}(\mathbf{y}_{0:K}) \cap X^m(\mathbf{y}_k)$. Since $X_{k|k-1}(\mathbf{y}_{0:K}) \subset \hat{X}_{k|k-1}$ by our inductive hypothesis, it follows that $X_{k|k}(\mathbf{y}_{0:K}) \subset \hat{X}_{k|k-1} \cap X^m(\mathbf{y}_k)$. Moreover, Assumption 4 implies that $X_{k|k}(\mathbf{y}_{0:K}) \subset G$, and hence $X_{k|k}(\mathbf{y}_{0:K}) \subset \hat{X}_{k|k-1} \cap X^m(\mathbf{y}_k) \cap G$. Then, by Condition 1 of Definition 4, $X_{k|k}(\mathbf{y}_{0:K}) \subset \mathcal{I}[\hat{X}_{k|k-1}, X^m(\mathbf{y}_k) \cap G] = \hat{X}_{k|k}$, as desired.

Next, we show that $X_{k+1|k}(\mathbf{y}_{0:K}) \subset \hat{X}_{k+1|k}$. Choose any $\mathbf{x}_{k+1} \in X_{k+1|k}(\mathbf{y}_{0:K})$. It suffices to show that that $\mathbf{x}_{k+1} \in \hat{X}_{k+1|k}$. By the definition of $X_{k+1|k}(\mathbf{y}_{0:K})$, there must exist $\mathbf{x}_k \in X_{k|k}(\mathbf{y}_{0:K})$ and $\mathbf{w}_k \in W$ such that $\mathbf{x}_{k+1} = \mathbf{x}_k + h\mathbf{f}(\mathbf{x}_k, \mathbf{w}_k)$. Choose any $i \in \{1, \dots, n_x\}$ and any

$$\mathbf{x}_k^* \in \operatorname{argmin}\left\{\|\mathbf{x} - \mathbf{x}_k\|_\infty : \mathbf{x} \in \Omega_i^L(\hat{X}_{k|k}, \mathbf{y}_k)\right\}. \tag{3.21}$$

Since f_i^L is a lower bounding function for f_i, it follows that $f_i^L\left(\Omega_i^L(\hat{X}_{k|k}, \mathbf{y}_k), W\right) \leq$

$f_i(\mathbf{x}_k^*, \mathbf{w}_k)$. Thus, we have

$$\begin{aligned}
&\hat{x}^L_{k+1|k,i} - x_{k+1,i} \\
&= \hat{x}^L_{k|k,i} - x_{k,i} + h\left(f_i^L(\Omega_i^L(\hat{X}_{k|k}, \mathbf{y}_k), W) - f_i(\mathbf{x}_k, \mathbf{w}_k)\right), \\
&\leq \hat{x}^L_{k|k,i} - x_{k,i} + h\left(f_i(\mathbf{x}_k^*, \mathbf{w}_k) - f_i(\mathbf{x}_k, \mathbf{w}_k)\right). \qquad (3.22)
\end{aligned}$$

In order to apply the Lipschitz condition on f_i, we now establish that the points $\mathbf{x}_k^*$ and $\mathbf{x}_k$ are both elements of $\bar{X}$. First, we have already shown that $\mathbf{x}_k \in X_{k|k}(\mathbf{y}_{0:K}) \subset \hat{X}_{k|k}$. Thus, the fact that $k \leq K^*$ implies that $\mathbf{x}_k \in \bar{X}$. Next, the definition of $\mathbf{x}_k^*$ implies that

$$\mathbf{x}_k^* \in \Omega_i^L(\hat{X}_{k|k}, \mathbf{y}_k) = \mathcal{I}[\beta_i^L(\hat{X}_{k|k}), X^m(\mathbf{y}_k) \cap G]. \qquad (3.23)$$

Thus, by Condition 1 of Definition 4, $\mathbf{x}_k^* \in \beta_i^L(\hat{X}_{k|k}) \subset \hat{X}_{k|k} \subset \bar{X}$, where the last inclusion again follows from the fact that $k \leq K^*$. Thus, (3.19) can be applied in (3.22) to obtain

$$\hat{x}^L_{k+1|k,i} - x_{k+1,i} \leq \hat{x}^L_{k|k,i} - x_{k,i} + hM\|\mathbf{x}_k^* - \mathbf{x}_k\|_\infty. \qquad (3.24)$$

We now show that $\|\mathbf{x}_k^* - \mathbf{x}_k\|_\infty$ is bounded above by $M_G(x_{k,i} - \hat{x}^L_{k|k,i})$. Define the interval $\hat{X}^i_{k|k}$ by setting $\hat{X}^i_{k|k,j} = \hat{X}_{k|k,j}$ for all $j \neq i$ and $\hat{X}^i_{k|k,i} = [x_{k,i}, x_{k,i}]$. Since $\mathbf{x}_k \in \hat{X}_{k|k}$, we have $\mathbf{x}_k \in \hat{X}^i_{k|k}$. Moreover, since $\mathbf{x}_k \in X_{k|k}(\mathbf{y}_{0:K}) \subset X^m(\mathbf{y}_k) \cap G$ by Assumption 4, it follows from Condition 1 of Definition 4 that $\mathbf{x}_k \in \mathcal{I}[\hat{X}^i_{k|k}, X^m(\mathbf{y}_k) \cap G]$. But, by definition, $\mathbf{x}_k^*$ is a point in $\mathcal{I}[\beta_i^L(\hat{X}_{k|k}), X^m(\mathbf{y}_k) \cap G]$ with the minimum possible infinity-norm distance from $\mathbf{x}_k$. Thus, it follows from the definition of the Hausdorff metric that

$$\|\mathbf{x}_k^* - \mathbf{x}_k\|_\infty \leq d_H(\mathcal{I}[\hat{X}^i_{k|k}, X^m(\mathbf{y}_k) \cap G], \tag{3.25}$$

$$\mathcal{I}[\beta_i^L(\hat{X}_{k|k}), X^m(\mathbf{y}_k) \cap G]). \tag{3.26}$$

Using the Lipschitz assumption on $\mathcal{I}[\cdot, X^m(\mathbf{y}_k) \cap G]$,

$$\|\mathbf{x}_k^* - \mathbf{x}_k\|_\infty \leq M_G d_H(\hat{X}^i_{k|k}, \beta_i^L(\hat{X}_{k|k})). \tag{3.27}$$

Noting that the intervals $\hat{X}^i_{k|k}$ and $\beta_i^L(\hat{X}_{k|k})$ differ only in their i^{th} components, it further follows that

$$\|\mathbf{x}_k^* - \mathbf{x}_k\|_\infty \leq M_G \left| x_{k,i} - \hat{x}^L_{k|k,i} \right|. \tag{3.28}$$

Now, since we have already shown that $\mathbf{x}_k \in X_{k|k}(\mathbf{y}_{0:K}) \subset \hat{X}_{k|k}$, we must have $\hat{x}^L_{k|k,i} - x_{k,i} \leq 0$. Then, (3.28) and (3.24) give

$$\hat{x}^L_{k+1|k,i} - x_{k+1,i} \leq \hat{x}^L_{k|k,i} - x_{k,i} + hMM_G \left| x_{k,i} - \hat{x}^L_{k|k,i} \right|, \tag{3.29}$$

$$= (1 - hMM_G)\left(\hat{x}^L_{k|k,i} - x_{k,i}\right). \tag{3.30}$$

Thus, the condition $h \in (0, \frac{1}{MM_G}]$ implies that $\hat{x}^L_{k+1|k,i} - x_{k+1,i} \leq 0$, and since the choice of i was arbitrary, we have $\hat{\mathbf{x}}^L_{k+1|k} - \mathbf{x}_{k+1} \leq \mathbf{0}$. The proof that $\hat{\mathbf{x}}^U_{k+1|k} - \mathbf{x}_{k+1} \geq \mathbf{0}$ is analogous. Thus, we have $\mathbf{x}_{k+1} \in \hat{X}_{k+1|k}$, as desired. □

It remains to define a specific refinement algorithm $\mathcal{I}[\cdot, X^m(\mathbf{y}_k) \cap G]$ and provide a bound for its Lipschitz constant M_G. For this purpose, we apply Algorithm 1 in [106], which is a slight modification of the algorithm originally given in Definition 4 of [24]. This algorithm defines a refinement $\mathcal{I}[\cdot, G]$ specifically for sets of the form $G \equiv \{\mathbf{x} \in X_{\text{nat}} : \mathbf{M}\mathbf{x} = \mathbf{b}\}$, where $\mathbf{M} \in \mathbb{R}^{m \times n_x}$, $\mathbf{b} \in \mathbb{R}^m$, and $X_{\text{nat}} \subset \mathbb{R}^{n_x}$ is an interval of *natural* bounds (i.e., non-negativity). For brevity, we do not repeat the algorithm here. However, given an argument $X \in \mathbb{IR}^{n_x}$, the basic idea is to refine X by first intersecting it with X_{nat}, and then considering rearrangements of the equations $\mathbf{M}\mathbf{x} = \mathbf{b}$ that isolate each x_j independently; i.e., $x_j = m_{ij}^{-1}(b_i - \sum_{l \neq j} m_{il} x_l)$. For every such rearrangement, taking the interval extension of the right-hand side potentially gives an improved bound for X_j. Theorem 4 in [106] shows that the implementation of this scheme given in Algorithm 1 there satisfies Definition 4 with the Lipschitz constant

$$M_G = [\max(\max_i \alpha_i, 1)]^Q, \tag{3.31}$$

where $\alpha_i \equiv (\|\mathbf{m}_i\|_1 / m_i^*) - 1$, $\mathbf{m}_i$ is the i^{th} row of $\mathbf{M}$, $m_i^* \equiv \min_j \{|m_{ij}| : |m_{ij}| > 0\}$, and Q is the number of iterations through all possible rearrangements of $\mathbf{M}\mathbf{x} = \mathbf{b}$.

To apply Algorithm 1 in [106] to refinements of the form $\mathcal{I}[\cdot, X^m(\mathbf{y}_k) \cap G]$ here, we must restrict our attention to the linear output equation

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{v}_k. \tag{3.32}$$

Then, Algorithm 1 can be applied to the set

$$G' \equiv \left\{ (\mathbf{x}, \mathbf{v}) \in X_{\text{nat}} \times V : \left[\begin{smallmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} \end{smallmatrix}\right] \left[\begin{smallmatrix} \mathbf{x} \\ \mathbf{v} \end{smallmatrix}\right] = \left[\begin{smallmatrix} \mathbf{b} \\ \mathbf{y}_k \end{smallmatrix}\right] \right\}. \tag{3.33}$$

Specifically, we define $\mathcal{I}[X, X^m(\mathbf{y}_k) \cap G] = \pi_{\mathbf{x}} \circ \mathcal{I}[X \times V, G']$, where $\pi_{\mathbf{x}}$ denotes the projection of the interval $\mathcal{I}[X \times V, G']$ onto its first n_x components. It is straightforward to show that this provides a valid refinement of X. Moreover, the Lipschitz constant for this refinement is bounded by (3.31) with $\left[\begin{smallmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} \end{smallmatrix}\right]$ in place of $\mathbf{M}$. Notably, this constant is independent of the value of $\mathbf{y}_k$, as required by Theorem 6.

Table 3.1: Definition of set-based state estimation methods of the form (3.15)–(3.18) compared in Examples 3.5.1 and 3.5.2.

	$\Omega_i^{L/U}(X, \mathbf{y})$	$\Omega(X, \mathbf{y})$
Method 1	$\mathcal{I}[\mathcal{B}_i^{L/U}(X)]$	$\mathcal{I}[X, X^m(\mathbf{y})]$
Method 2	$\mathcal{I}[\mathcal{B}_i^{L/U}(X), X^m(\mathbf{y})]$	$\mathcal{I}[X, X^m(\mathbf{y})]$
Method 3	$\mathcal{I}[\mathcal{B}_i^{L/U}(X), G]$	$\mathcal{I}[X, X^m(\mathbf{y}) \cap G]$
Method 4	$\mathcal{I}[\mathcal{B}_i^{L/U}(X), X^m(\mathbf{y}) \cap G]$	$\mathcal{I}[X, X^m(\mathbf{y}) \cap G]$

3.5 Numerical Results

This section compares our new state estimation method with five other methods. The first four methods are of the general form (3.15)–(3.18), but with different definitions of Ω and $\Omega_i^{L/U}$ than those given in (3.12)–(3.14), as described in Table 3.1. Method 4 is our new method, while Methods 1–3 are weaker methods that help to understand the key features of Method 4. We also compare against two common set-based state estimation algorithms based on zonotopes. These methods (Methods 5 and 6) are described in [23] and [22], respectively. However, we modify Method 6 to use the improved correction step described in [23]. Methods 5 and 6 are implemented with 10^{th} order zonotopes using the order reduction method in [22]. We report wall clock times for implementations in MATLAB R2015a on a Dell Precision T1700 with an i5-4690 CPU @ 3.50GHz and 16.0 GB RAM.

3.5.1 Example 1

The following discrete-time system describes a continuous-stirred tank reactor from [27], where x_i is the concentration (M) of species i:

$$\begin{aligned} x_{1,k+1} =& x_{1,k} + h\Big[- u_{3,k}x_{1,k}x_{2,k} - k_2 x_{1,k}x_{3,k} + \\ & \tau^{-1}\left(u_{1,k} - 2x_{1,k}\right)\Big], \\ x_{2,k+1} =& x_{2,k} + h\left[-u_{3,k}x_{1,k}x_{2,k} + \tau^{-1}\left(u_{2,k} - 2x_{2,k}\right)\right], \\ x_{3,k+1} =& x_{3,k} + h\left[u_{3,k}x_{1,k}x_{2,k} - k_2 x_{1,k}x_{3,k} - 2\tau^{-1}x_{3,k}\right], \\ x_{4,k+1} =& x_{4,k} + h\left[k_2 x_{1,k}x_{3,k} - 2\tau^{-1}x_{4,k}\right]. \end{aligned} \tag{3.34}$$

The parameters $\tau^{-1} = 0.05(\text{min}^{-1})$ and $k_2 = 0.4(\text{M}^{-1}\text{min}^{-1})$ are constant and $u_1 \in [0.9, 1.1]$ (M), $u_2 \in [0.8, 1.0]$ (M) and $u_3 \in [10, 50]$ (M) are time-varying uncertainties. The initial condition is $\mathbf{c}_0 = (0.036, 0.038, 0.36, 0.052)$ and is certain. The states x_2, x_3 and x_4 are measured, so that

$$\begin{aligned} y_{1,k} &= x_{2,k} + v_{1,k}, \\ y_{2,k} &= x_{3,k} + v_{2,k}, \\ y_{3,k} &= x_{4,k} + v_{3,k}, \end{aligned} \tag{3.35}$$

with $v_{1,k} \in [-10^{-2}, 10^{-2}]$ and $v_{2,k}, v_{3,k} \in [-10^{-3}, 10^{-3}]$.

We are not aware of any redundant algebraic relationships satisfied by the states of (3.34) that can be used to define a nontrivial set G. Therefore, we follow the approach in [26] to manufacture a set G by embedding (3.34) in a higher-dimensional system.

We first define the redundant states,

$$\begin{aligned}
z_{1,k} &= -\frac{1}{3}x_{1,k} - \frac{1}{3}x_{2,k} + \frac{1}{3}x_{3,k}, \\
z_{2,k} &= -\frac{1}{3}x_{1,k} - \frac{1}{3}x_{3,k} + \frac{1}{3}x_{4,k}, \\
z_{3,k} &= -x_{1,k} + 2x_{2,k} + x_{3,k}, \\
z_{4,k} &= x_{1,k} - x_{2,k} + x_{4,k}.
\end{aligned} \tag{3.36}$$

Next, we augment (3.34) with the corresponding difference equations,

$$\begin{aligned}
z_{1,k+1} &= z_{1,k} + h\Bigg[u_{3,k}x_{1,k}x_{2,k} - \frac{1}{3}\tau^{-1}\left(u_{1,k} + u_{2,k}\right) \\
&\quad - 2\tau^{-1}z_{1,k}\Bigg], \\
z_{2,k+1} &= z_{2,k} + h\left[k_2 x_{1,k}x_{3,k} - \frac{1}{3}\tau^{-1}u_{1,k} - 2\tau^{-1}z_{2,k}\right], \\
z_{3,k+1} &= z_{3,k} + h\tau^{-1}\left[2\left(u_{2,k} - z_{3,k}\right) - u_{1,k}\right], \\
z_{4,k+1} &= z_{4,k} + h\tau^{-1}\left(u_{1,k} - u_{2,k} - 2z_{4,k}\right).
\end{aligned} \tag{3.37}$$

The specific definitions of $\mathbf{z}_k$ above are chosen so that fortuitous term cancellations occur when deriving (3.37) from (3.34), which helps to mitigate overestimation when bounding the right-hand sides of (3.37) [26]. The solutions of this augmented system now satisfy (3.36) for all $k \in \mathbb{K}$ by design. Accordingly, we define $G \equiv \{(\mathbf{x}, \mathbf{z}) \in \mathbb{R}^8 : \mathbf{M}\left[\begin{smallmatrix}\mathbf{x}\\ \mathbf{z}\end{smallmatrix}\right] = \mathbf{b}\}$ with

$$\mathbf{M} = \begin{bmatrix} -1/3 & -1/3 & 1/3 & 0 & -1 & 0 & 0 & 0 \\ -1/3 & 0 & -1/3 & 1/3 & 0 & -1 & 0 & 0 \\ -1 & 2 & 1 & 0 & 0 & 0 & -1 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{3.38}$$

We define the refinement $\mathcal{I}[\cdot, X^m(\mathbf{y}_k) \cap G]$ and bound its Lipschitz constant as described in §3.4, which gives $M_G = 5$. To apply Thoerem 6, we are choosing $\bar{X} = [0, 0.13] \times [0, 0.13] \times [0, 0.45] \times [0, 0.5]$, which gives $M = 13.33$ by Theorem 4 in [106]. Thus, we choose $h = 0.015\text{s} \leq \frac{1}{MM_G}$ and $K = 600$.

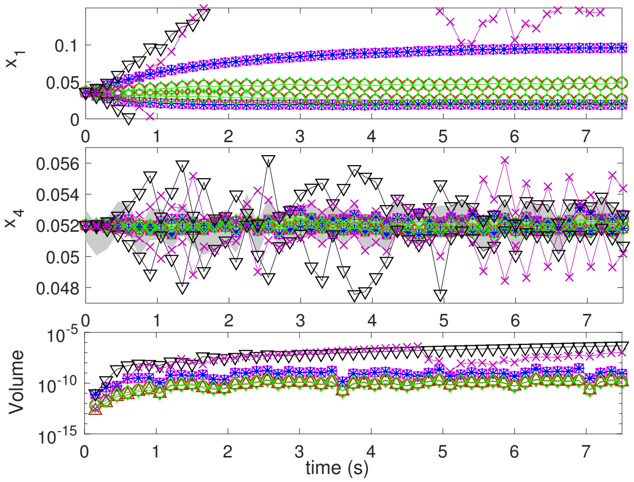

Figure 3.1: CPU time, volume, radius, and single-component projections of $\hat{X}_{k|k}$ for Methods 1–6 on Example 3.5.1 (□, ⋆, ○, ◇, ▽, ×), and volume and radius of $\hat{X}_{k|k} \cap G$ for Method 4 (△). Bounded error measurements are gray.

Figure 3.1 compares Methods 1–6 in terms of the required CPU time per time step k and the quality of the bound $\hat{X}_{k|k}$. Specifically, we compare the upper and lower bounds obtained by projecting $\hat{X}_{k|k}$ onto x_1 and x_4, as well as the volume and radius of $\hat{X}_{k|k}$. Note that x_4 is measured, as indicated by the gray shaded bounds, but x_1 is not. For Method 4, we also show the volume and radius of the polytope obtained by intersecting the interval $\hat{X}_{k|k}$ with G.

Figure 3.1 shows that our new method (◇) produces significantly tighter enclosures $\hat{X}_{k|k}$ than the state-of-the-art zonotope-based methods (▽, ×), and is also more efficient. Comparing the various DI methods in Table 3.1, we see that the methods making use of G (○, ◇) are more accurate but slightly more costly than those that do not use G (□, ⋆). However, there is no observable difference between the methods within these groups, which differ in the use of $X^m(\mathbf{y}_k)$ to refine $\beta_i^{L/U}(\hat{X}_{k|k})$ in the prediction step, as discussed immediately after (3.15)–(3.18). Interestingly, this results from the outputs $\mathbf{y}_k$ being simply a subset of the states $\mathbf{x}_k$, which makes $X^m(\mathbf{y}_k)$ an interval. In the next example, we show that there can be significant differences between these methods in the case of general linear outputs.

3.5.2 Example 2

Consider Example 3.5.1 with the outputs

$$
\begin{aligned}
y_{1,k} &= x_{1,k} + x_{2,k} + x_{3,k} + v_{1,k}, \\
y_{2,k} &= x_{2,k} + x_{3,k} + x_{4,k} + v_{2,k}, \\
y_{3,k} &= x_{1,k} + x_{4,k} + v_{3,k},
\end{aligned}
\tag{3.39}
$$

where $v_{1,k}, v_{2,k} \in [-10^{-2}, 10^{-2}]$ and $v_{3,k} \in [-10^{-3}, 10^{3}]$. In this case, $M_G = 5$ again, so the same step size h is valid.

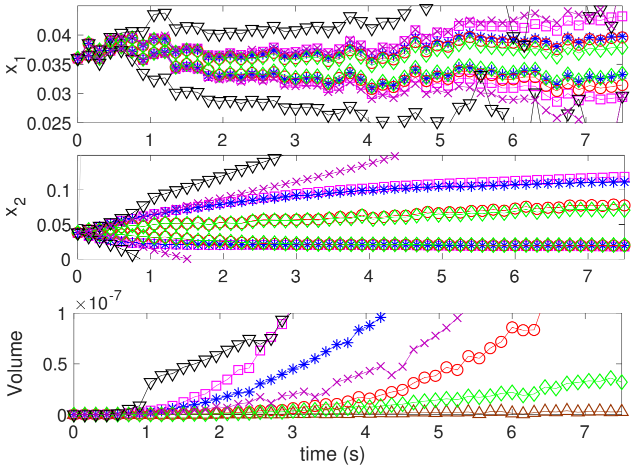

Figure 3.2: CPU time, volume, radius, and single-component projections of $\hat{X}_{k|k}$ for Methods 1–6 on Example 3.5.2 ($\square$, $\star$, $\circ$, $\diamond$, $\triangledown$, $\times$), and volume and radius of $\hat{X}_{k|k} \cap G$ for Method 4 ($\triangle$).

Figure 3.2 shows that our new method ($\diamond$) again produces significantly tighter enclosures $\hat{X}_{k|k}$ than the state-of-the-art zonotope-based methods ($\triangledown$, $\times$), and is again more efficient. Comparing the DI methods in Table 3.1, we also find again that the methods making use of G ($\circ$, $\diamond$) are more accurate but slightly more costly than those that do not use G ($\square$, $\star$). However, in this case, there is a clear difference between

Methods 1 and 2 ($\square$, $\star$), and also between Methods 3–4 ($\circ$, $\diamond$). These differences are particularly evident in the volume plot, and result from the use of $X^m(\mathbf{y}_k)$ to refine $\beta_i^{L/U}(\hat{X}_{k|k})$ in the prediction step of Methods 2 and 4, but not in Methods 1 and 3.

We draw the following general conclusions from these results. First, the state-of-the-art zonotope method in [23] consistently outperforms that in [22]. However, even the most basic DI method (Method 1) is more effective in all comparisons except volume. This is due to the use of the flattening operators $\beta_i^{L/U}$ permitted by the discrete-time DI theory developed here, which presently has no analogue in zonotope-based methods. The addition of model redundancy in Methods 3–4 relative to Methods 1–2 leads to much sharper enclosures with costs that are intermediate between Methods 1–2 and the zonotopic methods. Notably, Methods 3–4 significantly outperform the zonotopic methods even in terms of volume. Finally, in the case of general linear output equations, tighter enclosures are obtained by using $X^m(\mathbf{y}_k)$ to refine $\beta_i^{L/U}(\hat{X}_{k|k})$ in the prediction step of Methods 2 and 4 relative to Methods 1 and 3, which is a key new feature enabled by the DI estimation theory presented here.

3.6 Conclusion

This chapter proposed a novel approach for guaranteed state estimation using discrete-time DI method. The proposed state estimation algorithm largely increases the accuracy of the estimated state sets and is suitable for online applications. The main contribution of the chapter is exploiting the measurement set $X^m(\mathbf{y}_k)$ with DI method to refine for the prediction. The time step restriction for the new algorithm is stated in Section 3.3. The numerical results clearly verify the accuracy and efficiency of the proposed algorithm. The major limitation of the current algorithm is the lack of the refinement operator for nonlinear *a priori* set, which can be considered as the future work.

CHAPTER 4
GUARANTEED FAULT DETECTION USING DIFFERENTIAL INEQUALITIES

4.1 Introduction

This chapter applies the set-based state estimation method using differential inequalities in Chapter 3 to rapid and accurate set-based fault detection (FD) for highly nonlinear systems with uncertainties. Due to the level of complexity, integration, and automation in modern chemical processes, robotics systems, and power systems, faults such as equipment malfunctions and failures pose a serious threat to safe and profitable operation. Classical FD methods exploit historical data and are well established for various systems [41]. The methods detect faults by comparing observed measurements with previous statistics, which are often effective with sufficient historical data. However, these data-based methods cannot rigorously distinguish faults from system disturbances. This issue is particularly pronounced when systems have large uncertainties or when there is a lack of high-quality historical data that is relevant to the current operating conditions (e.g., an abnormal disturbance), which leads to false alarms and missed faults. An alternative class of FD methods exploits first-principles process models, which are available at least at the level of individual process units and subsystems in many applications of interest. In model-based approaches, faults are detected by comparing the process outputs that are consistent with the model (under all relevant uncertainties) to the outputs observed from the real process. Specifically, traditional model-based methods detect faults by checking if the difference between the predicted and measured outputs exceeds a threshold. However, the threshold value is usually empirical. Thus, choosing a threshold that minimizes missed faults without generating too many false alarms is challenging. Set-based FD is a particularly useful model-based approach that attempts to address this threshold problem rigorously. In set-based approaches, all uncertainties, disturbances, and measurement noises are assumed to be bounded and set-based computations are used to rigorously test if a new measured output is consistent with the process model given these bounds. This approach eliminates false alarms, but requires accurate set-based computations to achieve high sensitivity to faults, which is challenging.

Many set-based fault detection methods are available for linear systems using computations with intervals [42, 43], polytopes [44], ellipsoids [45], zonotopes [46, 47, 48], and constrained zonotopes [4]. However, testing the consistency of a measured output with a nonlinear model is significantly more difficult. One possible approach is to solve a nonlinear global optimization problem in each time step to determine if there exists a feasible point in the model that explains the current measurements. Although this would be accurate, it is clearly computationally intractable for most systems. A closely related idea was proposed in [49] for active input design rather than online fault detection. A second approach is to use set-based parameter estimation. In this approach, measurements are used to compute an enclosure of the set of model parameters that are consistent with the measurements, and a fault is detected when this enclosure has no overlap with a known set of possible parameter values for the fault-free model. The key challenge in this approach is how to compute tight enclosures of the feasible parameter set efficiently online. In [50], this is done using interval-based set inversion techniques. However, the computational cost scales exponentially with the number of uncertain parameters. This method is extended to systems with probabilistic noises using a Bayesian framework in [51]. However, this method does not provide rigorous bounds for uncertain parameters.

A third approach to set-based FD methods is to apply set-based state estimation. Recall that, in each time step, a set-based state estimator provides a guaranteed enclosure of the set of states consistent with the model, the bounded uncertainties, and all past measurements. This can then be used to compute an enclosure of the possible model outputs, and a fault is declared if the measured output is outside of this set. Note that some methods actually detect faults by computing a set of possible output prediction errors (i.e, residuals) rather than directly computing a set of possible outputs. As discussed in Chapter §3, the key challenge for these methods is to compute sufficiently accurate enclosures of the possible fault-free outputs (or residuals) fast enough for online fault detection. The articles [52] and [53] propose set-based FD approaches based on a Luenberger-type set-based state estimators. However, both methods compute rigorous enclosures of the residuals based on linear differential inclusions for the nonlinear observer error dynamics, which is likely to be very conservative for highly nonlinear systems. The article [54] also uses a Luenberger-type set-based state estimator. However, instead of computing a rigorous enclosure of all possible residuals for the fault-free model, they compute a smaller set of residuals based on a prescribed false alarm rate. Thus, this method is not guaranteed to avoid false alarms. Moreover, computing this set of residuals requires the solution of nonlinear chance constrained optimization problems in each time step, which is likely to be intractable for many systems. In order to reduce conservatism and increase efficiency, some approaches use approximate models with simpler structure. In [55], nonlinear models are linearized before constructing the observer, as in the extended Kalman filter. Similarly, the article [56] approximates nonlinear input-output models using a Takagi-Sugeno fuzzy neural network that is linear in the uncertain parameters. Rigorous ellipsoidal [56] and zonotopic [55] enclosures are then computed for the approximate models and used for fault detection. However, these enclosures are not rigorous for the original nonlinear systems and cannot provide guaranteed fault detection. Finally, the article [57] proposes a set-based fault detection method for continuous-time nonlinear systems based on enclosures of the fault-free states computed using advanced reachability techniques based on differential inequalities (DI). Although these reachability methods are very effective, they do not use measurements to refine the predicted enclosures as in a true set-based state estimator. Rather, measurements are only used to test for faults in each time step. This is a serious limitation and is likely to be prohibitive for systems with large uncertainties, where even the exact reachable set can be large.

To address these limitations, this Chapter develops a new set-based FD algorithm based on the set-based state estimation algorithm developed in Chapter 3. This algorithm guarantees no false alarms and improves upon the detection speed and fault sensitivity of existing set-based methods due to the superior accuracy and efficiency of our state estimator. The fault detection algorithm is firstly introduced in Section 4.2. The proposed algorithm is then compared with a popular data-based method based on principal component analysis (PCA), a conventional model-based method using the extended Kalman filter (EKF), and four state-of-the-art set-based algorithms. These algorithms are tested for four case studies and various scenarios within each case study, including fault-free cases normal disturbances, fault-free cases with large persistent disturbances, and cases with various faults. The results show that the proposed set-based algorithm eliminates false alarms and has the highest fault sensitivity among all set-based methods.

4.2 Set-Based Fault Detection Algorithm

Suppose that (3.1)–(3.2) represents the system model in the fault-free condition. Let $\mathbf{y}_{0:K} = (\mathbf{y}_0, \ldots, \mathbf{y}_K)$ be a measured output sequence and recall the sets $X_{k|k}(\mathbf{y}_{0:K})$ and $X_{k+1|k}(\mathbf{y}_{0:K})$ defined in (3.5)–(3.7). Furthermore, define the set

$$Y_{k|k-1}(\mathbf{y}_{0:K}) \equiv \{\mathbf{g}(\mathbf{x}, \mathbf{v}) : (\mathbf{x}, \mathbf{v}) \in X_{k|k-1}(\mathbf{y}_{0:K}) \times V\}. \tag{4.1}$$

In words, the set $Y_k(\mathbf{y}_{0:K})$ is the set of all outputs that can be generated by the nominal model (3.1)–(3.2) at time k given any inputs $\mathbf{x}_0 \in X_0$, $\mathbf{w}_{0:k-1} \in W_{0:k-1}$, and $\mathbf{v}_{0:k} \in V_{0:k}$ that could have generated the previous outputs $\mathbf{y}_{0:k-1}$.

Definition 5. A measured output sequence $\mathbf{y}_{0:K} = (\mathbf{y}_0, \ldots, \mathbf{y}_K)$ is said to be *inconsistent* with the model (3.1)–(3.2) if and only if $\mathbf{y}_k \notin Y_{k|k-1}(\mathbf{y}_{0:K})$.

According to Definition 5, if a measured output sequence becomes inconsistent, then the sequence can no longer be explained by the nominal model (3.1)–(3.2), and hence a fault must have occurred. The set-based fault detection approaches considered in this chapter aim to detect this situation as quickly as possible. On the other hand, it is assumed that no fault has occurred as long as the measured output remains consistent with the nominal model. The general structure of these methods is given in Algorithm 3, where $\hat{X}_{k|k}(\mathbf{y}_{0:K})$, $\hat{X}_{k+1|k}(\mathbf{y}_{0:K})$, and $\hat{Y}_{k|k-1}(\mathbf{y}_{0:K})$ are enclosures of $X_{k|k}(\mathbf{y}_{0:K})$, $X_{k+1|k}(\mathbf{y}_{0:K})$, and $Y_{k|k-1}(\mathbf{y}_{0:K})$ computed using a set-based state estimation algorithm of the general form given in (3.5)–(3.7). For brevity, we drop the argument $\mathbf{y}_{0:K}$ from these enclosures in Algorithm 3 and elsewhere where the output sequence is clear from context.

Algorithm 3 Set-based fault detection using a set-based state estimator

```
function FD($\mathbf{y}_{0:K}$, $C_0$, $W$, $V$)
    $\hat{X}_{0|-1} \supset C_0$
    for $k = 0$ to $K$ do
        $\hat{Y}_{k|k-1} \supset \{\mathbf{g}(\mathbf{x}, \mathbf{v}) : (\mathbf{x}, \mathbf{v}) \in \hat{X}_{k|k-1} \times V\}$
        if $\mathbf{y}_k \notin \hat{Y}_{k|k-1}$ then
            Generate an alarm and break
        end if
        $\hat{X}_{k|k} \supset \hat{X}_{k|k-1} \cap X^m(\mathbf{y}_k)$
        $\hat{X}_{k+1|k} \supset \{\mathbf{x} + h\mathbf{f}(\mathbf{x}, \mathbf{w}) : (\mathbf{x}, \mathbf{w}) \in \hat{X}_{k|k} \times W\}$
    end for
    return
end function
```

Lemma 1. *If Algorithm 3 is applied with input* ($\mathbf{y}_{0:K}$, C_0, W, V) *and, for some*

$k \in \{0, \ldots, K\}$, it happens that $\mathbf{y}_k \notin \hat{Y}_{k|k-1}$ in line 5, then $\mathbf{y}_{0:K}$ is inconsistent with (3.1)–(3.2).

Proof. Since $Y_{k|k-1} \subset \hat{Y}_{k|k-1}$ by definition, $\mathbf{y}_k \notin \hat{Y}_{k|k-1}$ indicates that $\mathbf{y}_k \notin Y_{k|k-1}$. Therefore, $\mathbf{y}_{0:K}$ is inconsistent with (3.1)–(3.2) by Definition 5. $\square$

The next lemma shows that the consistency test $\mathbf{y}_k \notin Y_{k|k-1}$ can be written equivalently in the state-space as $X_{k|k-1}(\mathbf{y}_{0:K}) \cap X^m(\mathbf{y}_k) = \emptyset$. This suggests that the enclosure-based test $\mathbf{y}_k \notin \hat{Y}_{k|k-1}$ used in Algorithm 3 could alternatively be replaced with the test $\hat{X}_{k|k-1}(\mathbf{y}_{0:K}) \cap X^m(\mathbf{y}_k) = \emptyset$, which may be advantageous in some cases. The next lemma gives a condition under which these two implementations are in fact equivalent.

Lemma 2. *For any measured output sequence $\mathbf{y}_{0:K} = (\mathbf{y}_0, \ldots, \mathbf{y}_K)$ and any $k \in \mathbb{K}$, $\mathbf{y}_k \notin Y_{k|k-1}(\mathbf{y}_{0:K})$ if and only if $X_{k|k-1}(\mathbf{y}_{0:K}) \cap X^m(\mathbf{y}_k) = \emptyset$. Moreover, if $\hat{Y}_{k|k-1}(\mathbf{y}_{0:K}) = \{\mathbf{g}(\mathbf{x}, \mathbf{v}) : (\mathbf{x}, \mathbf{v}) \in \hat{X}_{k|k-1}(\mathbf{y}_{0:K}) \times V\}$, then $\mathbf{y}_k \notin \hat{Y}_{k|k-1}(\mathbf{y}_{0:K})$ if and only if $\hat{X}_{k|k-1}(\mathbf{y}_{0:K}) \cap X^m(\mathbf{y}_k) = \emptyset$.*

Proof. Suppose that $\mathbf{y}_k \notin Y_k$. We prove that $X_{k|k-1} \cap X^m(\mathbf{y}_k) = \emptyset$ by contradiction. Suppose that there exists $\mathbf{x} \in X_{k|k-1} \cap X^m(\mathbf{y}_k)$. By the definition of $X^m(\mathbf{y}_k)$ in (3.4), it follows that there exist $\mathbf{v} \in V$ and $\mathbf{x} \in X_{k|k-1}$ such that $\mathbf{y}_k = \mathbf{g}(\mathbf{x}, \mathbf{v})$. But then $\mathbf{y}_k \in \{\mathbf{g}(\mathbf{x}, \mathbf{v}) : (\mathbf{x}, \mathbf{v}) \in X_{k|k-1} \times V\} = Y_k$, which is a contradiction.

Next suppose $X_{k|k-1} \cap X^m(\mathbf{y}_k) = \emptyset$. By the definition of $X^m(\mathbf{y}_k)$, $\mathbf{y}_k \neq \mathbf{g}(\mathbf{x}, \mathbf{v})$ for every $(\mathbf{x}, \mathbf{v}) \in X_{k|k-1} \times V$. Therefore, $\mathbf{y}_k \notin \{\mathbf{g}(\mathbf{x}, \mathbf{v}) : (\mathbf{x}, \mathbf{v}) \in X_{k|k-1} \times V\} = Y_k$.

To prove the second claim, suppose that $\mathbf{y}_k \notin \hat{Y}_k$. We prove that $\hat{X}_{k|k-1} \cap X^m(\mathbf{y}_k) = \emptyset$ by contradiction. Suppose that there exists $\mathbf{x} \in \hat{X}_{k|k-1} \cap X^m(\mathbf{y}_k)$. By the definition of $X^m(\mathbf{y}_k)$ in (3.4), it follows that there exist $\mathbf{v} \in V$ and $\mathbf{x} \in \hat{X}_{k|k-1}$ such that $\mathbf{y}_k = \mathbf{g}(\mathbf{x}, \mathbf{v})$. But then $\mathbf{y}_k \in \{\mathbf{g}(\mathbf{x}, \mathbf{v}) : (\mathbf{x}, \mathbf{v}) \in \hat{X}_{k|k-1} \times V\} = \hat{Y}_k$, which is a contradiction.

Next suppose $\hat{X}_{k|k-1} \cap X^m(\mathbf{y}_k) = \emptyset$. By the definition of $X^m(\mathbf{y}_k)$, $\mathbf{y}_k \neq \mathbf{g}(\mathbf{x}, \mathbf{v})$ for every $(\mathbf{x}, \mathbf{v}) \in \hat{X}_{k|k-1} \times V$. Therefore, $\mathbf{y}_k \notin \{\mathbf{g}(\mathbf{x}, \mathbf{v}) : (\mathbf{x}, \mathbf{v}) \in \hat{X}_{k|k-1} \times V\} = \hat{Y}_k$. $\square$

Although all of the set-based FD methods implemented in this chapter are based on the implementation in Algorithm 3 with the fault detection test $\mathbf{y}_k \notin \hat{Y}_{k|k-1}(\mathbf{y}_{0:K})$, it is more convenient to plot the numerical results in a way that shows the intersection $\hat{X}_{k|k-1}(\mathbf{y}_{0:K}) \cap X^m(\mathbf{y}_k)$. Since the condition $\hat{Y}_{k|k-1}(\mathbf{y}_{0:K}) = \{\mathbf{g}(\mathbf{x}, \mathbf{v}) : (\mathbf{x}, \mathbf{v}) \in \hat{X}_{k|k-1}(\mathbf{y}_{0:K}) \times V\}$ is satisfied in all of our examples, Lemma 2 ensures that these graphical representations are consistent with the way faults are detected in our algorithms.

4.2.1 Set-based Fault Detection Using Differential Inequalities

The new set-based FD algorithm proposed in the chapter results from implementing Algorithm 3 with the set-based state estimator developed in Chapter 3 using differential inequalities (DI). Specifically, we use the DI method (3.15)–(3.18) to compute $\hat{X}_{k|k}$ and $\hat{X}_{k+1|k}$ in lines 8 and 9:

$$\hat{X}_{k|k} = \Omega(\hat{X}_{k|k-1}, \mathbf{y}_k), \tag{4.2}$$

$$\hat{x}^L_{k+1|k,i} = \hat{x}^L_{k|k,i} + h f^L_i(\Omega^L_i(\hat{X}_{k|k}, \mathbf{y}_k), W), \tag{4.3}$$

$$\hat{x}^U_{k+1|k,i} = \hat{x}^U_{k|k,i} + h f^U_i(\Omega^U_i(\hat{X}_{k|k}, \mathbf{y}_k), W), \tag{4.4}$$

with $\Omega(X, \mathbf{y})$, $\Omega^L_i(X, \mathbf{y})$, and $\Omega^U_i(X, \mathbf{y})$ defined in Method 4 in Table 3.1.

To obtain the output enclosure $\hat{Y}_k$ required in line 4, we assume that an interval inclusion function $\mathcal{G} : \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_v} \to \mathbb{IR}^{n_y}$ is available for the measurement function $\mathbf{g}$ in (3.2), e.g., using interval arithmetic, and we denote $\mathcal{G}(X, V) = [\mathbf{g}^L(X, V), \mathbf{g}^U(X, V)]$.

Then, $\hat{Y}_k$ is obtained by

$$\begin{aligned} \hat{y}^L_{k,i} &= g^L_i(\hat{X}_{k|k-1}, V), \\ \hat{y}^U_{k,i} &= g^U_i(\hat{X}_{k|k-1}, V). \end{aligned} \tag{4.5}$$

4.3 Numerical Results

In this section, we compare seven fault detection methods using four numerical examples. For each example, the performance of each method is tested in multiple different faulty and fault-free scenarios. Performance is evaluated in terms of computational cost, the number of false alarms, and the time to detect a given fault using each method. We report wall clock times for implementations in MATLAB R2019b on a Macbook Pro with a 2.9 GHz Dual-Core Intel Core i5 processor and 8.0 GB RAM.

Method (i) is a conventional data-based method based on principal component analysis (PCA). PCA methods have been widely used for fault detection in industry [41]. Given a set of observation data, the PCA method first computes the eigendecomposition of the corresponding co-variance matrix. The eigenvalues indicate how much of the variance in the data is explained by each eigenvector. Any eigenvectors with small variance are eliminated, resulting in a remaining set of so-called principle directions. In many cases, this can lead to a substantial reduction in the dimensionality of the data. However, all of the case studies considered in this chapter are low dimensional systems and the computed eigenvalues indicated that the dimensionality could not be reduced further (i.e., all eigenvectors were retained). Next, the PCA method maps the data into a score space by computing the extent of each data point along each principle direction, normalized by the variance in that direction. Assuming that this data is normally distributed, a T^2 statistic is then used to compute a threshold containing the data with a specified probability level. This level is the probability that a new observation generated by the same dynamics will fall within the computed threshold

[41]. We compute the threshold by choosing a 95% probability level in all numerical examples as suggested by [41].

The assumption of normally distributed data in the PCA method is satisfied for linear systems with Gaussian noises. However, this assumption fails for nonlinear systems or systems with other kinds of uncertainties. As a result, the threshold computed by the PCA method is only approximate and cannot guarantee that a new observation falls within the threshold with 95% probability. Therefore, this method can potentially exhibit many false alarms for highly nonlinear systems.

Since we do not have real experimental data for the numerical examples considered here, we apply Method (i) using synthetic data generated by simulating 50000 observations starting from steady-state using the fault-free system dynamics and the distributions of uncertainties specified in each example.

Method (ii) is a model-based method that utilizes the extended Kalman filter (EKF) in Chapter 13 of [110]. We chose to compare with this EKF-based algorithm because the EKF is a very widely used state estimation technique in industry [111]. To use the EKF for fault detection, this method fist computes a set of residuals defined as the difference between the real measured outputs and the predicted outputs from the EKF. The method assumes that the system is fault-free if each residual sequence is an independent Gaussian random sequence with zero mean and a computed covariance. The system is faulty if the system follows an alternative hypothesis that residuals have positive or negative bias with mean a. As recommended in [112], we tested both $a = \pm 1$ and only gives the results with most false alarms rates. A sequential probability ratio test (SPRT) is performed to test if the statistics of the residuals follow this hypothesis. A threshold is computed for the likelihood ratio function based on a 5% probability of a false alarm and 5% probability of missing a fault that specifically follows the alternative hypothesis [112]. In the numerical results, we only show the SPRT for one output variable for each fault detection test. In fault-free cases, we show the variable with the most false alarms. In faulty cases, we show the variable that detects the fault earliest.

We also compared four set-based methods. All of these methods follow Algorithm 3, but each one uses a different set-based state estimator in lines 8–9. In Method (iii), state estimation is done using the DI method with invariants described in Chapter 3. More specifically, state estimation is done using (4.2)–(4.4) with $\Omega(X, \mathbf{y})$, $\Omega_i^L(X, \mathbf{y})$, and $\Omega_i^U(X, \mathbf{y})$ computed by Method 4 in Table 3.1. In Method (iv), state estimation is done using the standard DI method without invariants, which is described by (4.2)–(4.4) with $\Omega(X, \mathbf{y})$, $\Omega_i^L(X, \mathbf{y})$, and $\Omega_i^U(X, \mathbf{y})$ computed by Method 1 in Table 3.1. In Method (v), state estimation is done using a more basic discrete-time interval approach instead of DI. Specifically, the prediction step is done by taking the natural interval extension of (3.1) over the entire sets $\hat{X}_{k|k}$ and W, as in the so-called standard interval method in Chapter 2, and the correction step is done using (4.2) with $\Omega(X, \mathbf{y})$ computed by Method 1 in Table 3.1. In all of these interval methods, the computation of $\hat{Y}_k$ in line 4 of Algorithm 3 is done using (4.5) and, since $\hat{Y}_k$ is an interval, the condition $\mathbf{y}_k \notin \hat{Y}_k$ in line 5 is easily checked.

In Methods (vi) and (vii), state estimation is done using the zonotope-based methods described in Chapter 3 as Methods 6 and 5, respectively. Since all examples in this chapter have linear measurement equations of the form $\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{v}_k$, a zonotopic enclosure $\hat{Y}_k$ for use in line 4 can be computed as $\hat{Y}_k = \mathbf{C}\hat{X}_k \oplus V$. However, this would require solving a linear program to check $\mathbf{y}_k \notin \hat{Y}_k$ in line 5. To avoid this additional computational cost, we instead compute $\hat{Y}_k$ as the interval hull of $\mathbf{C}\hat{X}_k \oplus V$, where $\oplus$ is the Minkowski sum. Our numerical experiments show that using linear programming to detect faults in line (5) does not significantly increase the fault sensitivity of either method, but does increase the computational time by 5–6 times.

Methods (i)–(vii) are summarized in Table 4.1. These methods are compared in Figures 4.1–4.24 in the numerical example. The markers in the figures representing

Table 4.1: Summary descriptions and markers used for Methods (i)–(vii)

Method Index	**Method Description**	**Marker**
(i)	The data-based method using PCA	black lines
(ii)	The model-based method using EKF	blue lines
(iii)	The DI with invariants method	⋆
(iv)	The standard DI method	△
(v)	The standard interval method	□
(vi)	The zonotope method in [22]	◇
(vi)	The zonotope method in [23]	○

Methods (i)–(vii) and are also summarized in Table 4.1.

Note that Methods (i) and (ii) are derived assuming that all uncertain parameters are all normally distributed. On the other hand, Methods (iii)–(vii) assume that the uncertain parameters are bounded within compact sets with arbitrary distributions. In most of the following examples, we assume that the uncertain parameters obey normal distributions and we implement the set-based methods using the 99.7% confidence interval for each parameter. In a few cases, we assume that a parameter is interval bounded, in which case Methods (i) and (ii) are implemented using the normal distribution whose 99.7% confidence interval coincides with the specified bounds. Let $x \sim \mathcal{N}(\mu, \sigma)$ denote a normal distributed variable x with mean μ and standard deviation σ.

4.3.1 Example 1

Consider the continuous stirred-tank reactor (CSTR) with cooling from [107] in dimensionless form:

$$\begin{aligned} x_{1,k+1} &= x_{1,k} + \delta\left[w_{1,k} - x_{1,k} - D_a e^{-\frac{\alpha}{x_{2,k}}} x_{1,k}\right], \\ x_{2,k+1} &= x_{2,k} + \delta\Bigg[w_{2,k} - x_{2,k} - H D_a e^{-\frac{\alpha}{x_{2,k}}} x_{1,k} \\ &\qquad - S\left(x_{2,k} - \beta^{-1} x_{3,k}\right)\Bigg], \\ x_{3,k+1} &= x_{3,k} + \delta\left[\gamma(w_{3,k} - x_{3,k}) + S_c\left(\beta x_{2,k} - x_{3,k}\right)\right], \end{aligned} \tag{4.6}$$

where x_1, x_2, and x_3 are the dimensionless concentration, reactor temperature, and cooling water temperature. The time-varying disturbances are the dimensionless inlet concentration, reactor inlet temperature, and inlet cooling water temperature, which are denoted as $w_1 \sim \mathcal{N}(1, 0.2/3)$, $w_2 \sim \mathcal{N}(1, 0.84)$, and $w_3 \sim \mathcal{N}(1, 0.00195)$. The 99.7% confidence intervals used as interval bounds for the set-based methods are $w_1 \in [0.8, 1.2]$, $w_2 \in [0.9943, 1.006]$, and $w_3 \in [0.9929, 1.007]$. The dimensionless heat of reaction is $H = 0.5977$. The parameters $D_a = 4.93 \times 10^{11}$, $\alpha = 25$, $\beta = 1.2367$, $\gamma = 1.6096$, $S = 14.3291$, and $S_c = 4.0770$ are constant. The initial condition is $\mathbf{x}_0 = (0.0398, 0.96, 1.133)$, which is near the nominal steady state without disturbances.

There are no existing invariants for this systems. Therefore, in order to apply Method (iii), we manufacture invariants using the approach described in Chapter 2. First, we define the redundant state $z_k = -Hx_{1,k} + x_{2,k}$ and augment (4.6) with the redundant difference equation

$$z_{k+1} = z_k + \delta\Bigg[- H(w_{1,k} - x_{1,k}) + w_{2,k} - x_{2,k} - S\left(x_{2,k} - \beta^{-1} x_{3,k}\right)\Bigg]. \tag{4.7}$$

The solutions of the augmented system are now guaranteed to lie in the set

$$G \equiv \{(k, (\mathbf{x}, z)) \in \mathbb{K} \times \mathbb{R}^4 : \mathbf{m}_1 \left(\begin{smallmatrix} \mathbf{x} \\ z \end{smallmatrix} \right) = b_1,\ \mathbf{x} \geq \mathbf{0}\},$$

where $\mathbf{m}_1 = [-H \quad 1 \quad 0 \quad -1]$ and $b_1 = 0$. We choose time step $\delta = 0.001$ and total time $K = 300$. The dimensionless measurement equations are as follows, where $v_1, v_2 \sim \mathcal{N}(0, 0.01/3)$ and their 99.7% confidence intervals are $[-0.01, 0.01]$:

$$y_1 = x_2 + v_1,$$

$$y_2 = x_3 + v_2.$$

Methods (i)–(vi) are compared using the fault-free and faulty scenarios described in Table 4.2. The synthetic historical data needed for Method (i) was generated by simulating 100 trajectories over 500 time steps starting from the nominal steady-state $\mathbf{x}_0$ and with normally distributed uncertainties. For each scenario in Table 4.2, all the performance of all methods were compared using a new sequence of 'measured' outputs generated by simulating either the fault-free or faulty model with values of the uncertainties as described in the table.

The fault detection results are given in Figures 4.1–4.4. First, Figure 4.1 considers the fault-free scenario (a). The results show that the PCA method occasionally generates brief false alarms, while none of the other methods do. This is likely due to the fact that the system is nonlinear, which implies that the measured output data is not normally distributed even for normally distributed disturbances and measurement noises.

When there is a large persistent disturbance in the parameter w_2, the results in Figure 4.2 show that both PCA and EKF methods generate clear, persistent false alarms. In contrast, none of set-based methods gives a false alarm. This is the expected behavior of the PCA and EKF methods since such a persistent disturbance would be very unlikely if all uncertainties followed their typical distributions. Nonetheless, it shows a clear limitation of the PCA and EKF methods for handling problems with bounded uncertainties with unknown distribution.

In Scenario (c), a fault occurs at time 0.1 that causes the inlet concentration to decrease outside of its normal range. Figure 4.3 shows that both PCA and EKF methods detect the fault around time 0.13. Among all of the compared set-based methods, the DI method with invariants detects the fault earliest, around time 0.2. The standard DI method (iv) also detects the fault, but more slowly. In contrast, Methods (v)–(vii) all fail to detect this fault at all due to overly conservative state estimation sets.

Finally, Figure 4.4 shows the results in Scenario (d), in which a fault occurs at time 0.1 that causes the cooling temperature increase outside of its normal range. PCA, EKF, and the DI methods (iii) and (iv) all detect the fault at around the same time after 0.1. The zonotope methods also detect the fault in this case, but more slowly than the DI methods.

In this example, all uncertainties are normally distributed and we used their 99.7% confidence intervals as the interval bounds for the set-based methods (iii)–(vii). This assumption favors Methods (i) and (ii) and puts all of the set-based methods at a disadvantage. If the uncertainties do not follow normal distributions in practice, Methods (i) and (ii) will not be as effective as shown in Scenario (a). This is the reason that Methods (i) and (ii) generate persistent false alarms in Scenario (b). However, these two methods detect faults rapidly. In contrast, the set-based methods (iii)–(vii) do not assume any distribution for the uncertainties and guarantee no false alarms. The DI method (iii) detects faults the earliest among all set-based methods, and is nearly as fast as Methods (i) and (ii) for Scenario (d), although about 4× slower in Scenario (c). Finally, the DI with invariants algorithm only takes 0.027 s of CPU time for every time step of the algorithm, which is about 5× faster than the the zonotope

Table 4.2: Faulty and fault-free scenarios for Example 4.3.1. The uncertainties w_1–w_3 and v_1–v_2 are time-varying, independent, and normally distributed unless specified otherwise.

Scenario Index	**Fault-Free Scenario Descriptions**
(a)	All uncertain parameters are as in the caption.
(b)	At time 0.1, w_2 takes a large constant value within its 99.7% confidence interval $w_2 \in [0.9943, 1.006]$: $w_2 = 1$.
	Faulty Scenario Descriptions
(c)	At time 0.1, the inlet concentration decreases to the constant value $w_1 = 0.7$, which is outside its confidence interval $w_1 \in [0.8, 1.2]$.
(d)	At time 0.1, cooling temperature increases to the constant value $w_3 = 290/283$, which is outside its confidence interval $w_3 \in [0.9929, 1.007]$.

methods.

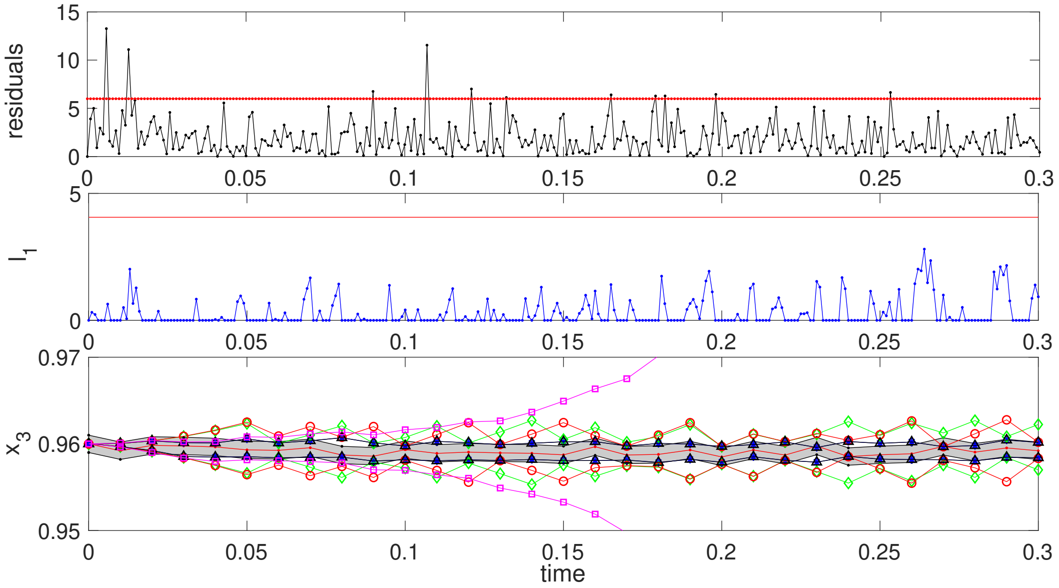

Figure 4.1: Fault detection results for Scenario (a) in Example 4.3.1 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds ($\star$, $\triangle$, $\square$, $\diamond$, $\circ$) have empty intersection with the bounded-error measurement (gray shaded).

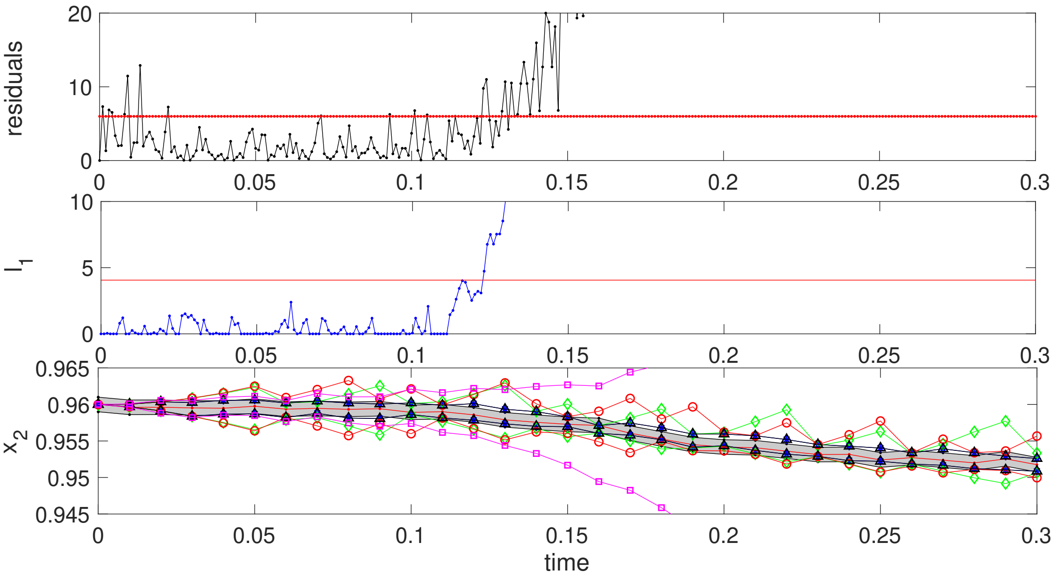

Figure 4.2: Fault detection results for Scenario (b) in Example 4.3.1 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds (⋆, △, □, ◇, ○) have empty intersection with the bounded-error measurement (gray shaded).

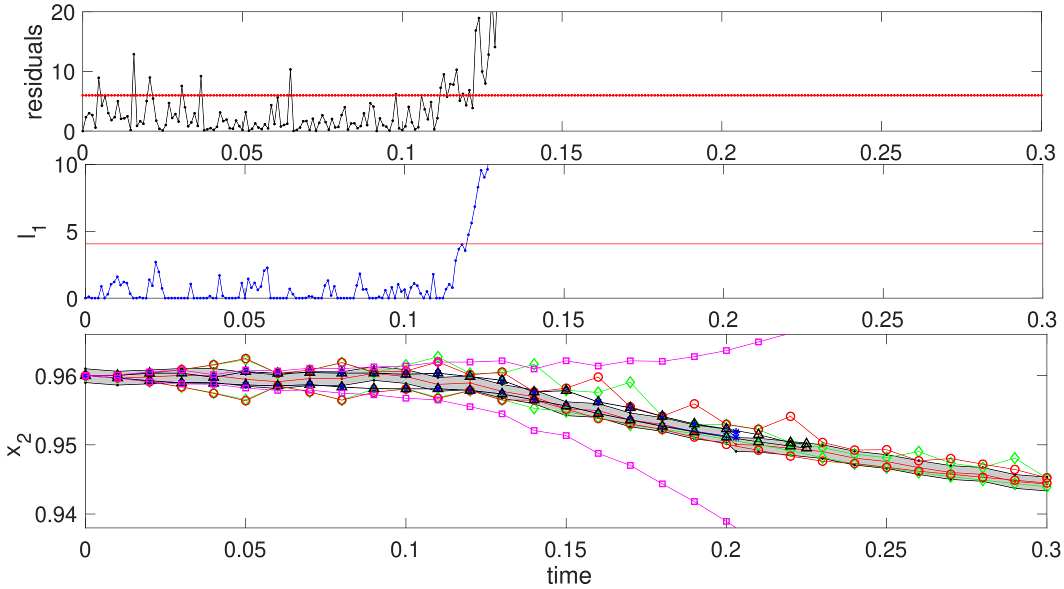

Figure 4.3: Fault detection results for Scenario (c) in Example 4.3.1 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds (⋆, △, □, ◇, ○) have empty intersection with the bounded-error measurement (gray shaded).

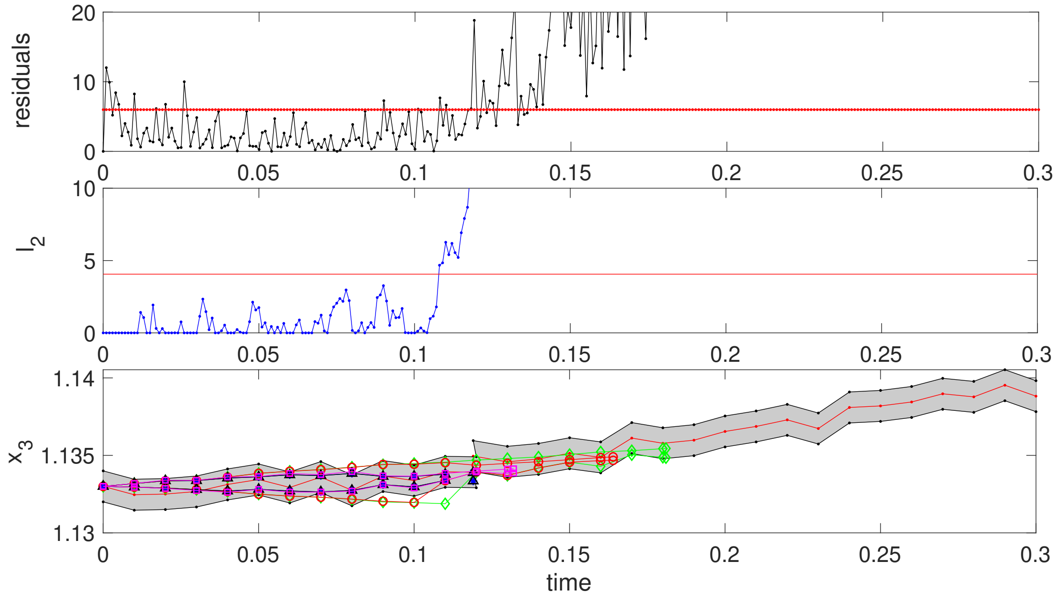

Figure 4.4: Fault detection results for Scenario (d) in Example 4.3.1 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds ($\star$, $\triangle$, $\square$, $\diamond$, $\circ$) have empty intersection with the bounded-error measurement (gray shaded).

4.3.2 Example 2

The following dynamics describe a sewer system with three tanks, where x_i is the water volume (m^3) of tank i and u_i is the i^{th} inlet flow rate (m^3/s) [8]:

$$
\begin{aligned}
x_{1,k+1} &= x_{1,k} + h\left[u_{1,k} + u_{2,k} - \kappa_1 x_{1,k}\right] \qquad (4.8)\\
x_{2,k+1} &= x_{2,k} + h\left[\kappa_1 x_{1,k} - \kappa_2\sqrt{x_{2,k}}\right]\\
x_{3,k+1} &= x_{3,k} + h\left[\kappa_2\sqrt{x_{2,k}} + u_{3,k} - \kappa_3 x_{3,k}\right]
\end{aligned}
$$

We set $u_i = d_i + w_i$ with $\mathbf{d} = (1, 2, 1)$ and disturbances $w_i \sim \mathcal{N}(0, 0.1/3)$. The time-invariant parameters κ_i are uncertain but bounded: $\kappa_1 \in [4.8, 6.8] \times 10^{-4}$,

$\kappa_2 \in [1.99, 2.01] \times 10^{-2}$, and $\kappa_3 \in [9.9, 10.1] \times 10^{-4}$. The measurement functions are

$$y_1 = x_2 + v_1,$$

$$y_2 = x_3 + v_2,$$

where the measurement noises $v_1 \sim \mathcal{N}(0, 100/3)$ and $v_2 \sim \mathcal{N}(0, 50/3)$ are uncertain and time-varying. The initial condition $\mathbf{x}_0 = (5400, 23000, 4000)$ is near the fault-free nominal steady-state and is certain.

Synthetic historical data was generated for Method (i) by simulating 10 trajectories of the fault-free system over 5000 time steps with the uncertain parameters w_1–w_3 and v_1–v_2 normally distributed as described above. In each of these ten trajectories, a different sample of the time-invariant uncertainties κ_1–κ_3 was randomly drawn from a uniform distribution over the given bounds. This was intended to mimic a situation where κ_1–κ_3 are physical parameters that are time-invariant on the time-scale of operation but may vary over longer time-scales; i.e., due to changes in environmental conditions. Method (ii) assumes that all uncertainties are time-varying and normally distributed, and the mean and covariance of these uncertainties are used explicitly in the computations. Therefore, there is no satisfactory way to accommodate time-invariant uncertainties like κ_1–κ_3 within Method (ii). For lack of a better option, we implemented Method (ii) using independent normal distributions for κ_1–κ_3 formed by considering their interval bounds as their 99.7% confidence intervals: $\kappa_1 \sim \mathcal{N}(0.0005, 0.0001/3)$, $\kappa_2 \sim \mathcal{N}(0.02, 0.0001/3)$, and $\kappa_3 \sim \mathcal{N}(0.001, 0.00001/3)$.

All of the set-based methods are implemented assuming that each normally distributed uncertain parameter is bounded within its 99.7% confidence interval: $w_i \in [-0.1, 0.1]$, $v_1 \in [-100, 100]$ and $v_2 \in [-50, 50]$. To apply Method (iii), the manufactured invariants given in Example 2.6.2 in Chapter 2 were used. Finally, we chose a step size of $h = 30$ s and total a number of time steps $K = 500$.

Table 4.3: Faulty and fault-free scenarios for Example 4.3.2. Unless otherwise specified, the uncertainties w_1–w_3 and v_1–v_2 are time-varying, independent, and normally distributed. In Scenarios (b)–(c), the time-invariant uncertainties κ_1–κ_3 are sampled from independent uniform distributions at time 0. In Scenarios (d)–(g), κ_1–κ_3 are fixed to values near the center of their bounding intervals in order to reduce false alarms in Method (ii) and more clearly show the effects of the faults in those scenarios.

Scenario Index	**Fault-Free Scenario Descriptions**
(a)	All uncertain parameters are as in the caption.
(b)	w_1–w_3 and v_1–v_2 are as in the caption. $\kappa_1 - \kappa_3$ are time-varying, independent, and normally distributed.
(c)	The initial condition is perturbed away from steady-state by 10% to $\mathbf{x}_0 = (5000, 20000, 3500)$.
(d)	At 3000 s, w_1 takes a large constant value within its 99.7% confidence interval $[-0.1, 0.1]$: $w_1 = 0.08$.
Scenario Index	**Faulty Scenario Descriptions**
(e)	At 3000 s, the flow rates increase to constant values outside their 99.7% confidence intervals $[-0.1, 0.1]$: $w_1 = w_2 = 0.15$ and $w_3 = 0.2$.
(f)	At 3000 s, the second tank begins leaking. The dynamics change to: $x_2 = x_{2,k} + h\left[\kappa_1 x_{1,k} - (\kappa_2 + 0.005)\sqrt{x_{2,k}}\right]$.
(g)	At 3000 s, a sensor fault happens causing the constant noise value $v_2 = 100$, which is outside the confidence interval $[-50, 50]$.

All methods were compared in all of the fault-free and faulty scenarios described in Table 4.3. Scenario (a) is the nominal case as discussed above. The results are shown in Figure 4.5. Because the EKF method assumes that κ_1–κ_3 are normally distributed, it generates many false alarms in this normal condition. Although the PCA method is also derived assuming that the data are normally distributed, PCA is much more robust in this case because it was trained on synthetic historical data generated by the true distributions.

In order to investigate the cause of the poor performance of Method (ii) in Scenario (a), Scenario (b) considers a fault-free case where the 'measured' outputs are actually generated with κ_1–κ_3 normally distributed and time-varying. The results in Figure 4.6 show that the number of false alarms for the EKF method is greatly reduced as expected. For this scenario, the PCA method was also retrained using historical data generated by the modified κ_1–κ_3, and again exhibited no false alarms. This experiment confirms that the EKF method is not fit to handle problems with time-invariant bounded uncertainties and should be expected to give frequent false alarms in such cases. On the other hand, the set-based methods (iii)–(vi) have no false alarms in either Scenario (a) or (b).

Scenario (c) considers a fault-free case where the system starts from an initial condition perturbed away from the fault-free steady state. Figure 4.7 shows that the data-based method (i) generates a clear and persistent false alarm immediately, clearly mistaking the initial transient for a fault. The EKF method also generates a clear false alarms at early times. The results of a similar scenario tested in Example 4.3.3 show that Method (ii) using EKF deals with different initial conditions very well. Thus, the false alarms triggered by EKF in this example may be caused by the same reason as in Scenario (a). In contrast, the set-based methods again exhibit no false alarms. This illustrates another key advantage of these methods. Because they make use of a dynamic process model, the are able to clearly distinguish the effects of the initial transient from the effects of a fault.

In scenarios (d)–(g), the time-invariant parameters are fixed to values near their mean values: $\kappa_1 = 5 \times 10^{-4}$, $\kappa_2 = 2 \times 10^{-2}$, and $\kappa_3 = 10^{-3}$. This was done in order to avoid excessive false alarms in Method (ii) and more clearly illustrate the disturbances and faults considered in these scenarios. Note that in practice $\kappa_1 - \kappa_3$ may not take values near their means, in which case the EKF method will generate many false alarms and not be useful for fault detection in scenarios (d)–(g).

Figure 4.8 shows the results of Scenario (d), where a large, persistent disturbance occurs in w_1, but there is no fault. In this case, the EKF method generates aclear false alarm. The residual for the PCA method trends upwards, but does not result in a false alarm, while the set-based methods again exhinit no false alarms.

The results of the faulty scenarios (e)–(g) are shown in Figures 4.9–4.11. In Scenario (e), both the PCA and EKF methods detect the fault immediately after it happens. The set-based methods (iii) and (iv) using DI and the zonotope method (vii) detect the fault about 1000 s later than PCA and EKF. The zonotope method (vi) detects the fault shortly after Methods (iii),(iv), and (vii). Finally, the standard interval method is not sensitive to detect this fault. In Scenario (f), an extra term is added to κ_2 to simulate a leak in Tank 2. The results show that only PCA, EKF, and Method (iii) using DI with invariants can detect this fault, where Method (iii) detects the fault 2000 s later than the other two methods. In Scenario (g), the results show that all the methods including all set-based methods are very sensitive to the sensor fault in this scenario.

In this example, we assume that some of the uncertain parameters are time-varying and normally distributed and some of the uncertainties are time-invariant and bounded in given intervals. Since Method (ii) is strongly dependent on the assumption of time-varying normal uncertainties, it generates false alarms in all of the fault-free scenarios. Moreover, since the data-based method (i) only applies to steady-state data, it becomes ineffective during process transients. Furthermore, a sensor bias as a new fault detection scenario in this example can be detected immediately by the set-based methods (iii)–(vii), which are as fast as Methods (i) and (ii). Finally, the DI with invariants is very efficient and only takes 0.001 s to compute for every time step of the algorithm. In contrast, the computational cost of the zonotope methods are more than 3 times higher than the DI method (iii).

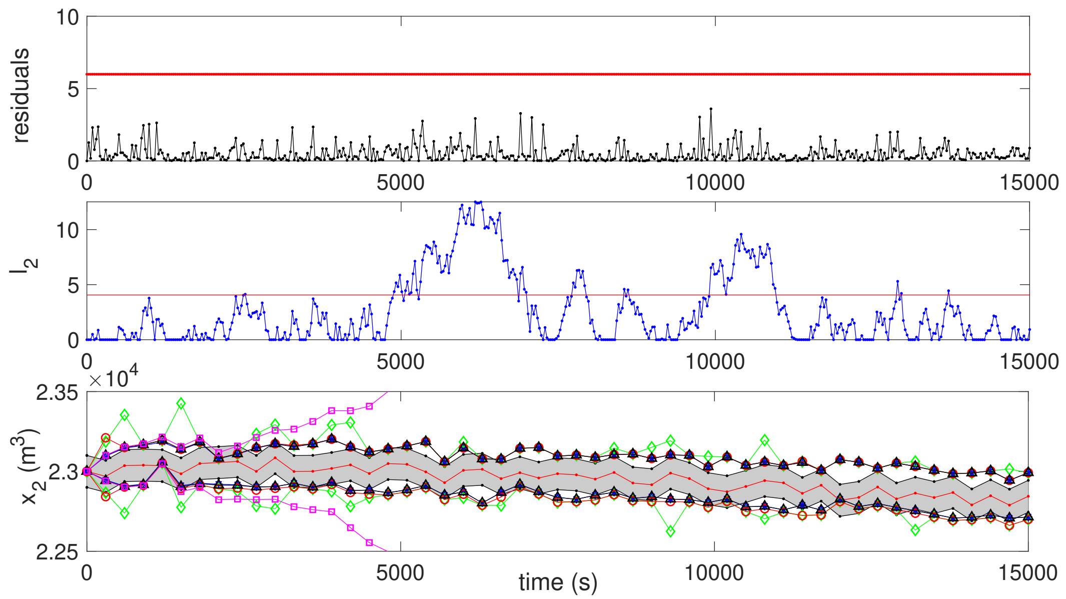

Figure 4.5: Fault detection results for Scenario (a) in Example 4.3.2 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds (⋆, △, □, ◇, ○) have empty intersection with the bounded-error measurement (gray shaded).

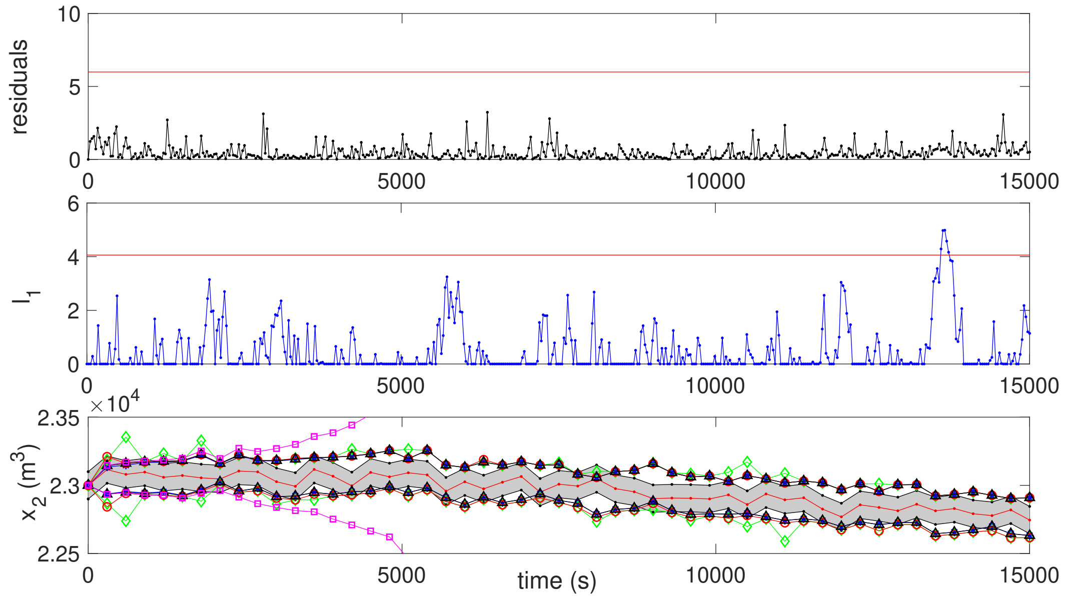

Figure 4.6: Fault detection results for Scenario (b) in Example 4.3.2 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds (⋆, △, □, ◇, ○) have empty intersection with the bounded-error measurement (gray shaded).

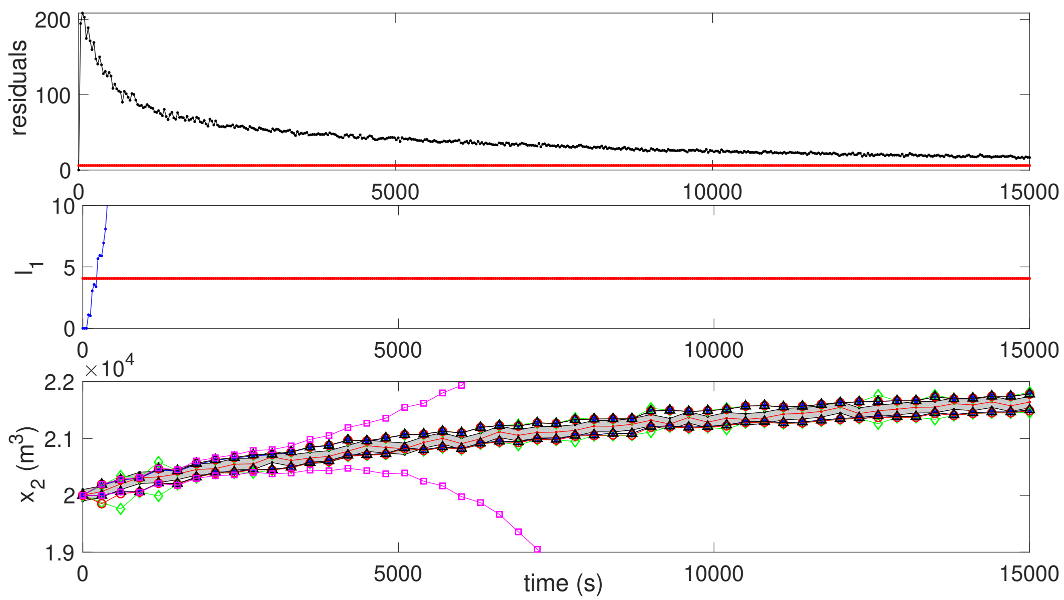

Figure 4.7: Fault detection results for Scenario (c) in Example 4.3.2 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds (⋆, △, □, ◇, ○) have empty intersection with the bounded-error measurement (gray shaded).

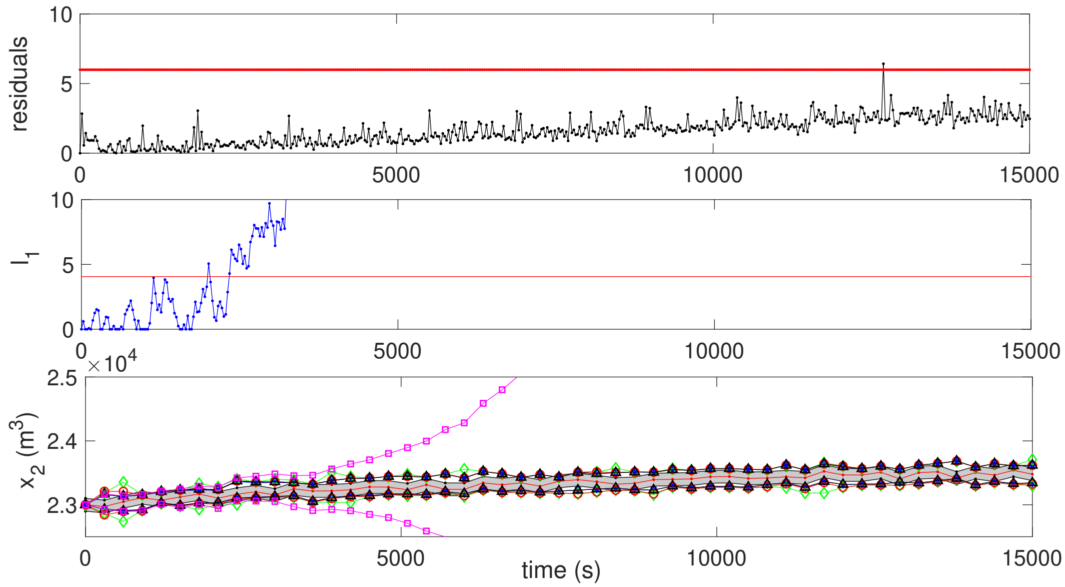

Figure 4.8: Fault detection results for Scenario (d) in Example 4.3.2 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds (⋆, △, □, ◇, ○) have empty intersection with the bounded-error measurement (gray shaded).

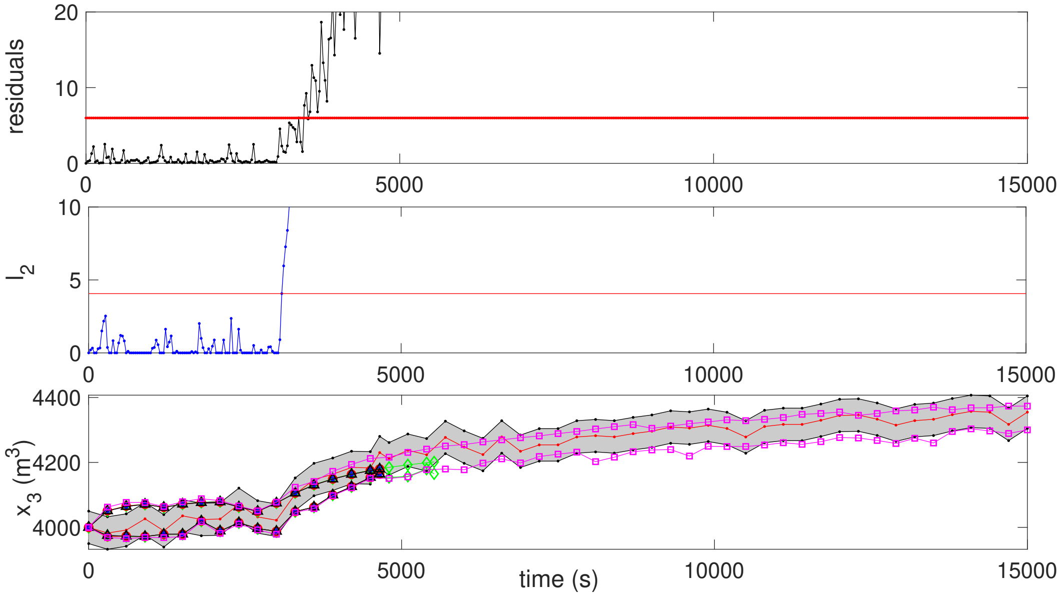

Figure 4.9: Fault detection results for Scenario (e) in Example 4.3.2 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds ($\star$, $\triangle$, $\square$, $\diamond$, $\circ$) have empty intersection with the bounded-error measurement (gray shaded).

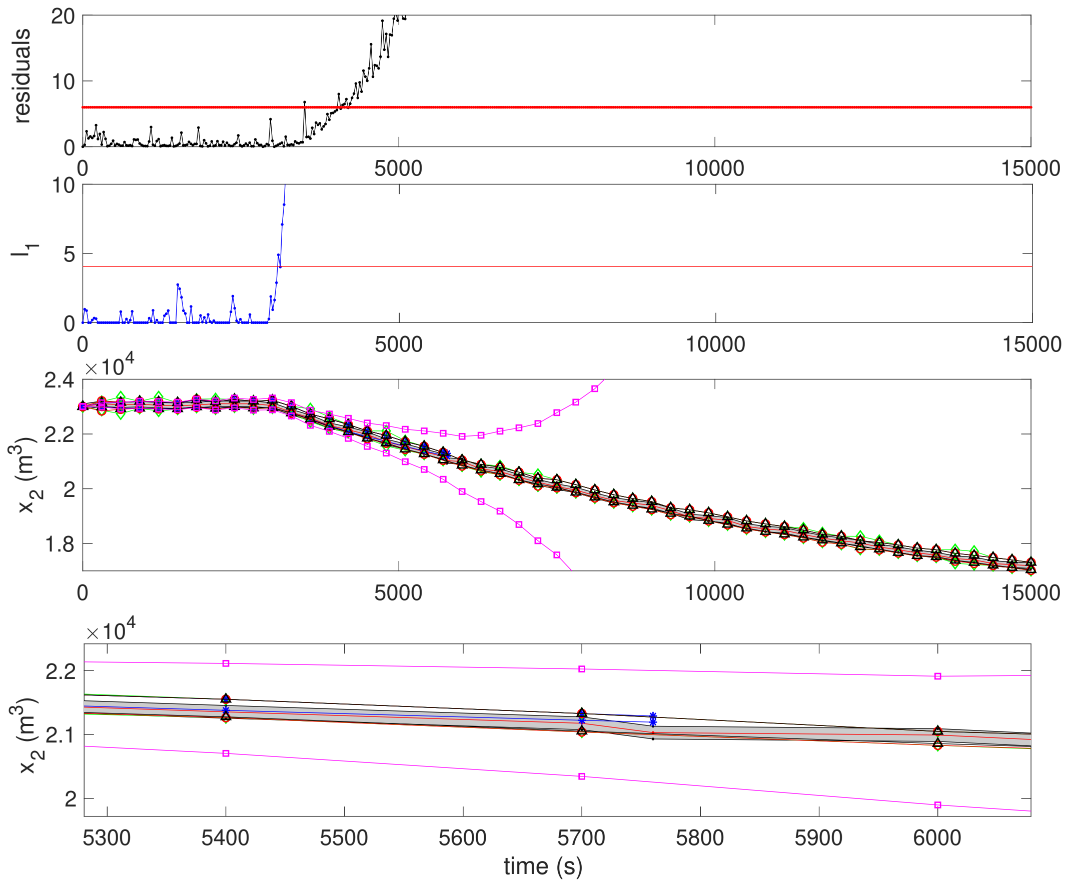

Figure 4.10: Fault detection results for Scenario (f) in Example 4.3.2 using Method (i) (top), Method (ii) (the second), Methods (iii)-(vii) (the third), and zoomed-in Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds (⋆, △, □, ◇, ○) have empty intersection with the bounded-error measurement (gray shaded).

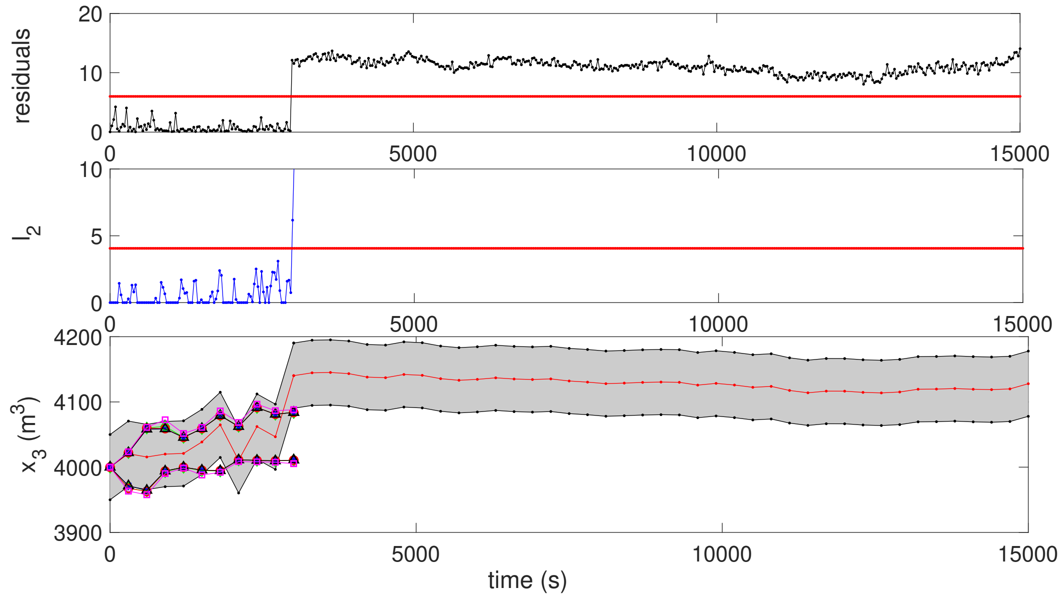

Figure 4.11: Fault detection results for Scenario (g) in Example 4.3.2 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds (⋆, △, □, ◇, ○) have empty intersection with the bounded-error measurement (gray shaded).

4.3.3 Example 3

Consider the following model of a CSTR from [26], where x_i is the concentration (M) of species i, the parameters $\tau^{-1} = 0.05(\text{min}^{-1})$ and $k_2 = 0.4(\text{M}^{-1}\text{min}^{-1})$ are constant, and $u_1 \sim \mathcal{N}(1, 0.1/3)$ (M), $u_2 \sim \mathcal{N}(0.9, 0.1/3)$ (M), and $u_3 \sim \mathcal{N}(30, 20/3)$ (M) are time-varying uncertainties:

$$
\begin{aligned}
x_{1,k+1} &= x_{1,k} + h\Big[-u_{3,k}x_{1,k}x_{2,k} - k_2 x_{1,k}x_{3,k} + \tau^{-1}\left(u_{1,k} - 2x_{1,k}\right)\Big] \qquad (4.9)\\
x_{2,k+1} &= x_{2,k} + h\left[-u_{3,k}x_{1,k}x_{2,k} + \tau^{-1}\left(u_{2,k} - 2x_{2,k}\right)\right]\\
x_{3,k+1} &= x_{3,k} + h\left[u_{3,k}x_{1,k}x_{2,k} - k_2 x_{1,k}x_{3,k} - 2\tau^{-1}x_{3,k}\right]\\
x_{4,k+1} &= x_{4,k} + h\left[k_2 x_{1,k}x_{3,k} - 2\tau^{-1}x_{4,k}\right]
\end{aligned}
$$

We choose the time step $h = 0.015$ min and the horizon $K = 800$. The measurement equations are

$$\begin{aligned} y_{1,k} &= x_{2,k} + v_1, \\ y_{2,k} &= x_{3,k} + v_2, \\ y_{3,k} &= x_{4,k} + v_3, \end{aligned} \tag{4.10}$$

with measurement noises $v_1 \sim \mathcal{N}(0, 0.01/3)$ and $v_2, v_3 \sim \mathcal{N}(0, 0.001/3)$. The initial condition is chosen as $\mathbf{x}_0 = (0.036, 0.038, 0.36, 0.052)$, which is close to the fault-free steady-state.

To apply Method (i) using PCA, synthetic historical data was generated by simulating 100 trajectories of the fault-free model. In all set-based methods, the uncertainties were assumed to be bounded within their 99.7% confidence intervals: $u_1 \in [0.9, 1.1]$ (M), $u_2 \in [0.8, 1.0]$ (M), $u_3 \in [10, 50]$ (M), $v_1 \in [-0.01, 0.01]$, $v_2 \in [-0.001, 0.001]$, and $v_3 \in [-0.001, 0.001]$. In order to apply Method (iii), the manufactured invariants developed in Example 3.5.1 were used.

Six fault detection scenarios are given in Table 4.4. The results for each scenario are shown in Figures 4.12–4.17.

Scenario (a) is the nominal case. Figure 4.13 shows that both PCA and EKF generate a modest number of false alarms in this case. In order to investigate the impact of non-Gaussian distribution on these methods, Scenario (b) considers all uncertainties to be uniformly distributed within the interval bounds given above. The PCA method was retrained specifically for Scenario (b) using synthetic historical data generated with these modified distributions. Figure 4.13 shows that both PCA and EKF exhibit many more false alarms in this case, which illustrates again the shortcomings of these methods for systems with non-Gaussian uncertainties.

In Scenario (c), the initial condition is perturbed by 10% away from the fault-free

Table 4.4: Faulty and fault-free scenarios for Example 4.3.3. Unless otherwise specified, the uncertainties u_1–u_3 and v_1–v_3 are time-varying, independent, and normally distributed.

Scenario Index	**Fault-Free Scenario Descriptions**
(a)	All uncertain parameters are as in the caption.
(b)	u_1–u_3 and v_1–v_3 are time-varying, independent, and uniformly distributed in the given bounds.
(c)	The initial condition is perturbed away from the fault-free steady-state by 10% to $\mathbf{x}_0 = (0.032, 0.034, 0.32, 0.047)$.
(d)	At 4.5 min, u_1–u_3 take large constant values within their 99.7% confidence intervals: $u_1 = 1.05$, $u_2 = 0.95$, and $u_3 = 45$.
Scenario Index	**Faulty Scenario Descriptions**
(e)	At 4.5 min, the inlet concentration decreases to a constant value outside its 99.7% confidence interval: $u_1 = 0.5$.
(f)	At 4.5 min, the residence time decreases by 40% to $\tau = 12$.

nominal steady state. Figure 4.14 shows that the PCA method generates a clear false alarm after a short time and becomes ineffective. This shows once again that the PCA method cannot distinguish between process transients and faults because it is trained on steady-state data. In contrast, all methods that make use of the dynamic process model are able to make this distinction clearly, including all set-based methods as well as the EKF method. While the EKF method does generate some false alarms in this scenario, the number of false alarms is similar to Scenario (a).

In Scenario (d), a large persistent disturbance happens at 4.5 min but no fault occurs. The results in Figure 4.15 show that both PCA and EKF generate false alarms shortly after 4.5 min and become ineffective. In contrast, the set-based methods give no false alarms.

In all of the previuos scenarios, Methods (vi) and (vii), which use zonotopic set-based state estimators, both exhibit strange oscillatory prediction bounds. This is correct and is attributable to to some technical details of the heuristics used for approximating the intersection in the correction step of these methods.

In Scenario (e), a fault occurs at 4.5 min that causes the inlet flow rate to take a constant value outside its normal range. Figure 4.16 shows that PCA, EKF, and DI with invariants are the only methods able to detect this fault. PCA detects the fault at 5 min, which is a half minute earlier than the EKF and DI with invariants.

In Scenario (f), a fault occurs at 4.5 min that causes the residence time of the reactor to change to a different value due to channeling. Figure 4.17 shows that PCA detects the fault immediately after it occurs, followed by EKF and DI with invariants about 1 minute later.

The computational time of Method (iii) is 0.0031 s, while the zonotope methods take about 0.0085 s to compute for every time step.

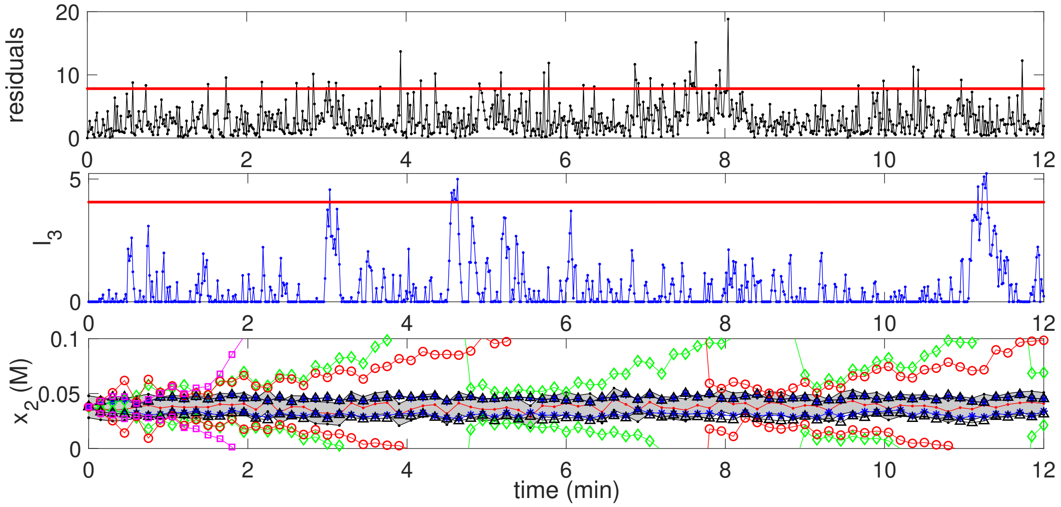

Figure 4.12: Fault detection results for Scenario (a) in Example 4.3.3 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds (⋆, △, □, ◇, ○) have empty intersection with the bounded-error measurement (gray shaded).

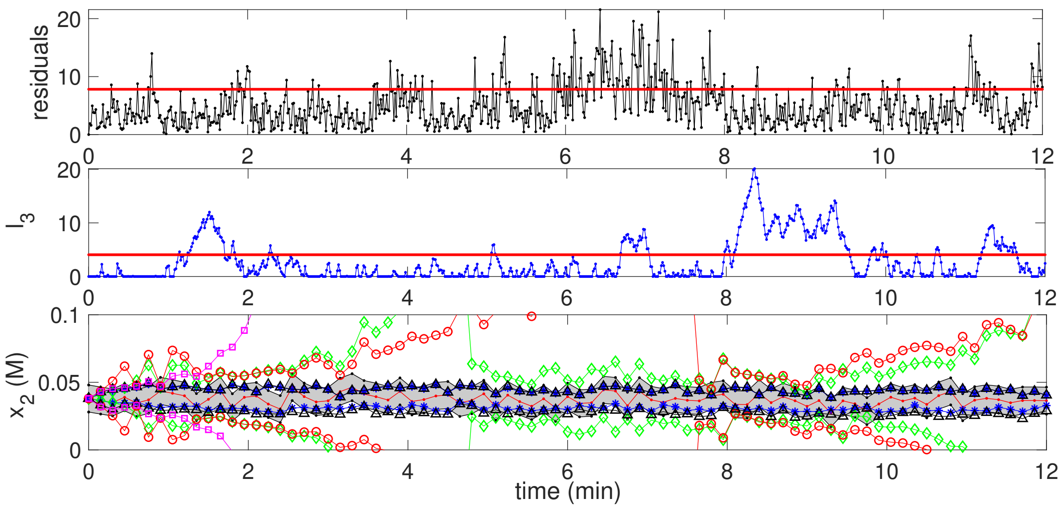

Figure 4.13: Fault detection results for Scenario (b) in Example 4.3.3 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds (⋆, △, □, ◇, ○) have empty intersection with the bounded-error measurement (gray shaded).

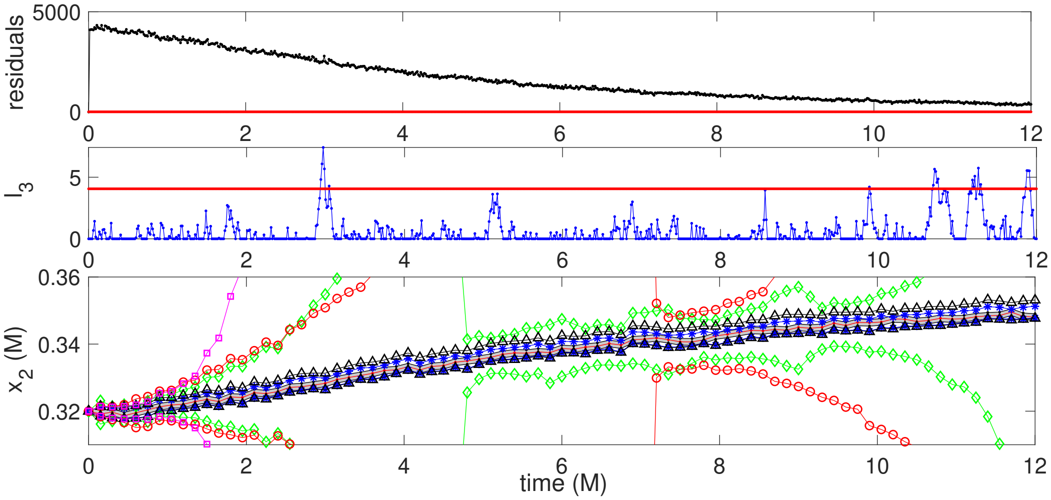

Figure 4.14: Fault detection results for Scenario (c) in Example 4.3.3 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds (⋆, △, □, ◇, ○) have empty intersection with the bounded-error measurement (gray shaded).

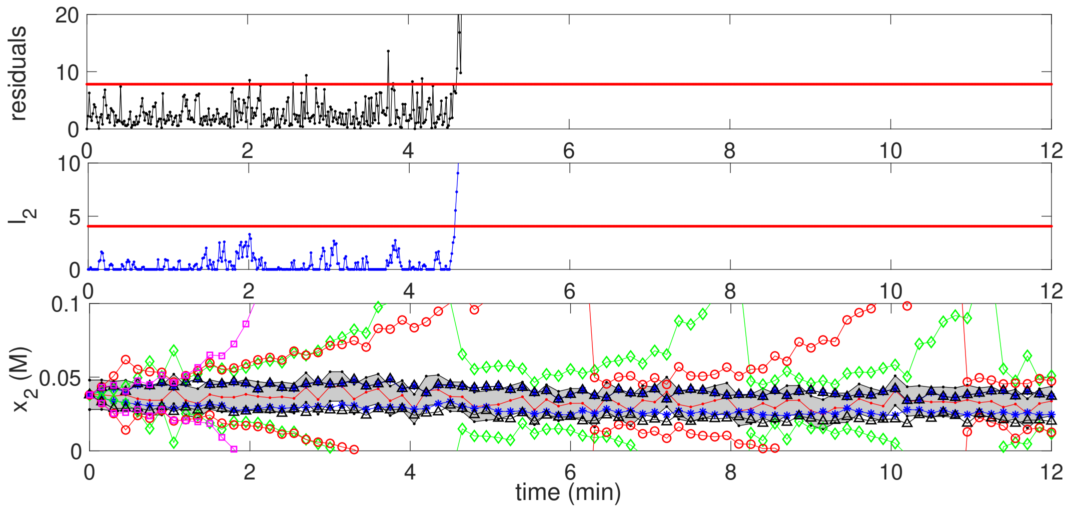

Figure 4.15: Fault detection results for Scenario (d) in Example 4.3.3 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds ($\star$, $\triangle$, $\square$, $\diamond$, $\circ$) have empty intersection with the bounded-error measurement (gray shaded).

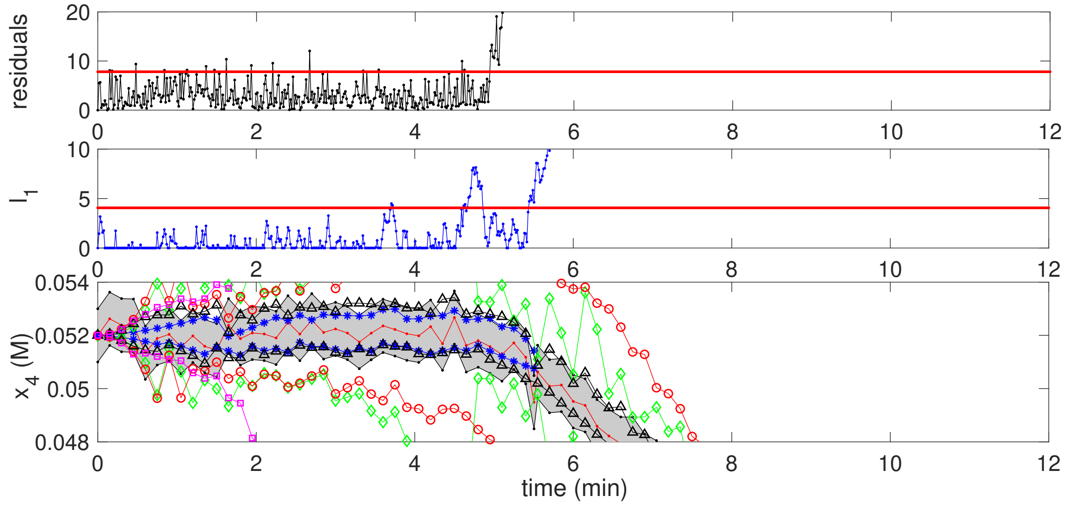

Figure 4.16: Fault detection results for Scenario (e) in Example 4.3.3 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds ($\star$, $\triangle$, $\square$, $\diamond$, $\circ$) have empty intersection with the bounded-error measurement (gray shaded).

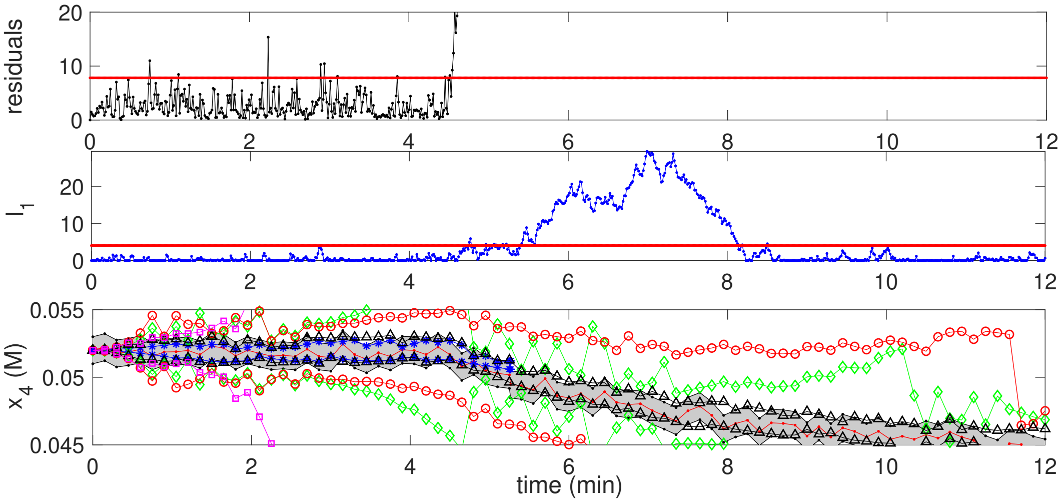

Figure 4.17: Fault detection results for Scenario (f) in Example 4.3.3 using Method (i) (top), Method (ii) (middle), and Methods (iii)-(vii) (bottom). Methods (i) and (ii) declare a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds ($\star$, $\triangle$, $\square$, $\diamond$, $\circ$) have empty intersection with the bounded-error measurement (gray shaded).

4.3.4 Example 4

The following dynamics describe an enzymatic reaction network with six chemical species occurring in a batch reactor, where x_i is the concentration (M) of species i: [24]:

$$
\begin{aligned}
x_{1,k+1} &= x_{1,k} + h\left[-k_{1,k}x_{1,k}x_{2,k} + k_{2,k}x_{3,k} + k_{6,k}x_{6,k}\right] \qquad (4.11)\\
x_{2,k+1} &= x_{2,k} + h\left[-k_{1,k}x_{1,k}x_{2,k} + k_{2,k}x_{3,k} + k_{3,k}x_{3,k}\right]\\
x_{3,k+1} &= x_{3,k} + h\left[k_{1,k}x_{1,k}x_{2,k} - k_{2,k}x_{3,k} - k_{3,k}x_{3,k}\right]\\
x_{4,k+1} &= x_{4,k} + h\left[k_{3,k}x_{3,k} - k_{4,k}x_{4,k}x_{5,k} + k_{5,k}x_{6,k}\right]\\
x_{5,k+1} &= x_{5,k} + h\left[-k_{4,k}x_{4,k}x_{5,k} + k_{5,k}x_{6,k} + k_{6,k}x_{6,k}\right]\\
x_{6,k+1} &= x_{6,k} + h\left[k_{4,k}x_{4,k}x_{5,k} - k_{5,k}x_{6,k} - k_{6,k}x_{6,k}\right].
\end{aligned}
$$

Let $\hat{\mathbf{k}} = (0.1, 0.033, 16, 5, 0.5, 0.3)$. The parameters $\mathbf{k} = (k_1, \ldots, k_6)$ are uncertain and time-invariant with $\mathbf{k} \in [\hat{\mathbf{k}}, 10\hat{\mathbf{k}}]$. Every state is measured, so $y_i = x_i + v_i$ with

$v_i \sim \mathcal{N}(0, 0.1/3)$ for all i. The initial condition is $\mathbf{x}_0 = (34, 20, 0, 0, 16, 0)$ and is certain.

Method (i) is not compared in this problem because the PCA method only applies to steady-state processes and this is a non-steady batch reactor model. Method (ii) is implemented assuming that $k_i \sim \mathcal{N}(5.5k_i, 1.5k_i)$ for all i. These distributions have the prescribed bounds $[\hat{\mathbf{k}}, 10\hat{\mathbf{k}}]$ as their 99.7% confidence intervals. To apply Method (iii), we manufacture invariants as given in Example 2.6.1 and assume $v_i \in [-0.1, 0.1]$ for all set-based methods, which is the 99.7% confidence interval for v_i.

The fault-free and faulty scenarios tested are described in Table 4.5. Scenario (a) is the nominal case. Figure 4.18 shows that Method (ii) generates consistent false alarms after 0.001s. As discussed in Example 4.3.2, the EKF method is based on the assumption that all parameters are normally distributed and produces many false alarms for other distributions. In contrast, the set-based methods (iii)–(vi) are very flexible with respect to the distributions of uncertainties and do not exhibit false alarms.

In Scenario (b), all of the uncertainties are modifed to follow time-varying normal distributions, as assumed by Method (ii). As expected, Figure 4.19 shows that Method (ii) is performs much better in this case. None of the tested methods generate any false alarms.

In Scenario (c), k_3 takes a large constant value within its normal range. Figure 4.20 shows that the EKF method generates many false alarms in this case, while the set based methods do not.

Scenarios (d)–(g) are all faulty scenarios. For all of these scenarios, the time-invariant parameters k_1–k_6 are set to constant values near their means rather than randomly sampled at time zero. This is done to avoid false alarms in the EKF method simply due to k_1–k_6 being time-invariant and non-Gaussian so that the effects of faults can be seen more clearly.

In Scenario (d), a sensor fault occurs that changes the distribution of v_3. The fault is immediately detected by all set-based methods. The EKF method also generates alarms, but there is not a clear and persistent fault signal, so it is hard to tell if the alarms are triggered by disturbances or the real fault.

Scenarios (e) and (f) represent cases where enzyme deactivation causes one of the reaction rate constants to drop outside of its normal range. The fault in Scenarios (e) is detected immediately by the EKF method, but is not detected by any of the set-based methods. The fault in Scenario (f) is detected early by the EKF method and the DI methods (iii) and (iv), but is not detected by the other set-based methods. Further investigation into Scenario (e) shows that this fault does not change the dynamics by much, and should therefore be considered a difficult fault to detect. Thus, $\mathbf{y}_k$ remains in the set $\hat{Y}_k$ predicted by all set-based methods. In fact, it may be true that $\mathbf{y}_k$ even remains within the exact set of consistent outputs, indicating that there is some combination of admissible uncertainties that would result in the same trajectory that is generated by the fault, although this is difficult to verify. In such cases, passive set-based fault detection methods will be unable to detect the fault and active fault detection is needed, which we leave for future work.

Scenario (g) represents a that occurs during loading of the batch reactor and causes an incorrect initial state of the batch reactor. In this case, the state estimators used in all fault detection methods are initialized with the desired initial state, while the real trajectory producing the output measurements is initialized from the faulty initial state. Note that this change in initial condition is considered a fault in Example 4, while similar situations were considered fault-free scenarios in earlier examples. The difference is that, in all earlier examples, the scenarios with modified initial conditions represented non-faulty cases where the process was deliberately and knowingly operated from a different initial state, with the initial states of all state estimators changed accordingly. Therefore, the trajectories in those cases were consistent with the models used for fault detection. In contrast, in this scenario, the estimators are not aware of the incorrect initial condition and the trajectory is not consistent with the fault-free model.

To make Scenario (g) non-trivial, we only measure x_1, x_3, and x_6, so that the faulty initial values of x_2 and x_5 are not immediately detectable from the measurement at the initial time. We also choose the time horizon as $K = 1000$. Figure 4.24 shows that the set-based method using DI with invariants (Method (iii)) is the only method that detects the initial condition fault. This is in part due to the use of invariants in Method (iii) that depend on the initial condition. Since the fault initial condition in this scenario violates the invariants used on Method (iii), the method is able to detect the fault effectively.

Table 4.5: Faulty and fault-free scenarios of Example 4.3.4. Unless otherwise specified, the uncertain parameters v_1–v_6 are time-varying, independent, and normally distributed and k_1–k_6 are time-invariant and uniformly distributed. In Scenarios (c)–(g), k_1–k_6 are fixed to the following values near their means rather than randomly sampled: $k_1 = 0.55$, $k_2 = 0.18$, $k_3 = 88$, $k_4 = 27$, $k_5 = 2.7$, and $k_6 = 1.6$.

Scenario Index	**Fault-Free Scenario Descriptions**
(a)	All uncertainties are as in the caption.
(b)	k_1–k_6 are time-varying, independent, and normally distributed.
(c)	k_3 takes a large constant value within its normal range $k_3 \in [16, 160]$: $k_3 = 145$.
	Faulty Scenario Descriptions
(d)	A sensor fault happens: Measurement noise v_3 changes to a uniform distribution $v_3 \in [-0.15, 0.15]$, which is larger than its 99.7% confidence interval $[-0.1, 0.1]$ of its original distribution.
(e)	Enzyme deactivates: $k_3 = 8$, which is outside its normal range $k_3 \in [16, 160]$.
(f)	Enzyme deactivates: $k_1 = 0.08$, which is outside its normal range $k_1 \in [0.1, 1]$.
(g)	The initial condition of the batch is faulty: $\mathbf{x} = (34, 25, 0, 0, 20, 0)$.

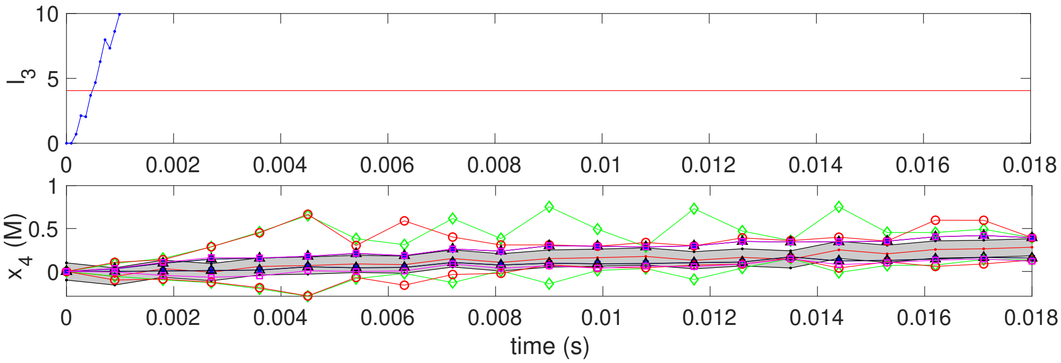

Figure 4.18: Fault detection results for Scenario (a) in Example 4.3.4 using Method (ii) (top) and Methods (iii)-(vii) (bottom). Methods (ii) declares a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds ($\star$, $\triangle$, $\square$, $\diamond$, $\circ$) have empty intersection with the bounded-error measurement (gray shaded).

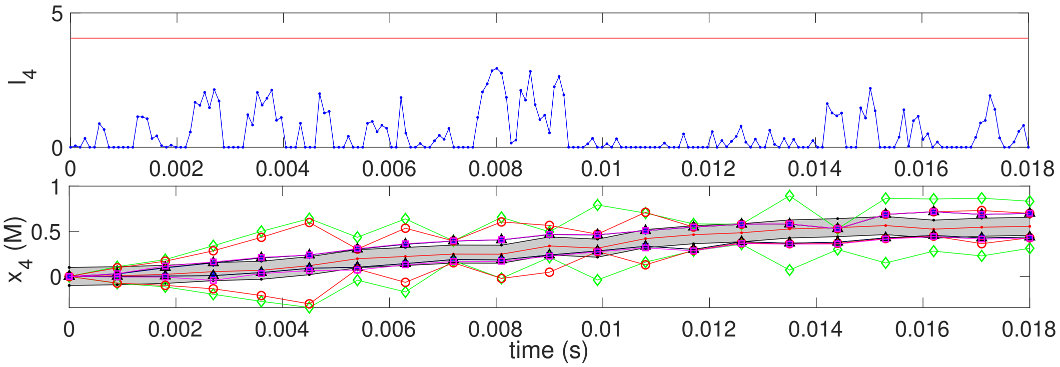

Figure 4.19: Fault detection results for Scenario (b) in Example 4.3.4 using Method (ii) (top) and Methods (iii)-(vii) (bottom). Methods (ii) declares a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds ($\star$, $\triangle$, $\square$, $\diamond$, $\circ$) have empty intersection with the bounded-error measurement (gray shaded).

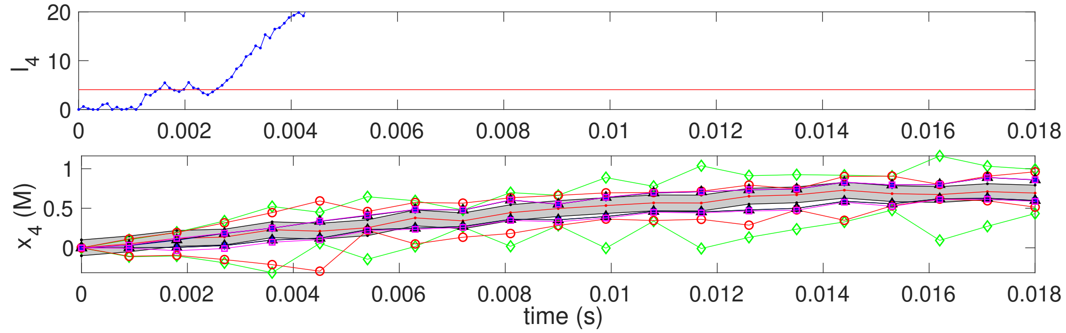

Figure 4.20: Fault detection results for Scenario (c) in Example 4.3.4 using Method (ii) (top) and Methods (iii)-(vii) (bottom). Methods (ii) declares a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds (⋆, △, □, ◇, ○) have empty intersection with the bounded-error measurement (gray shaded).

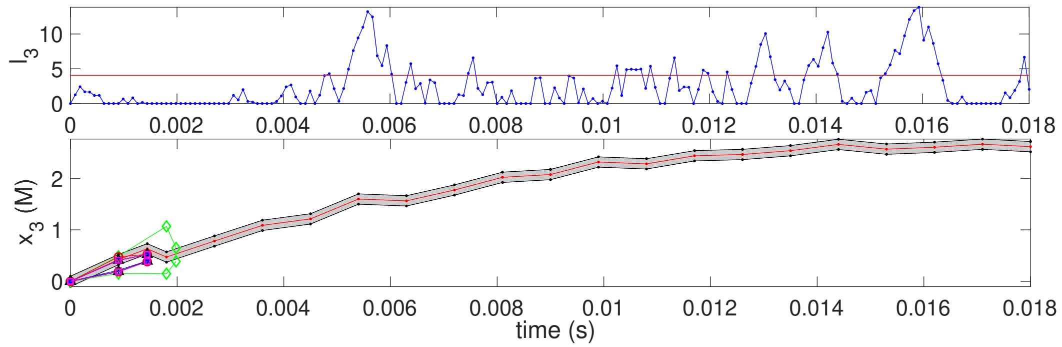

Figure 4.21: Fault detection results for Scenario (d) in Example 4.3.4 using Method (ii) (top) and Methods (iii)-(vii) (bottom). Methods (ii) declares a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds (⋆, △, □, ◇, ○) have empty intersection with the bounded-error measurement (gray shaded).

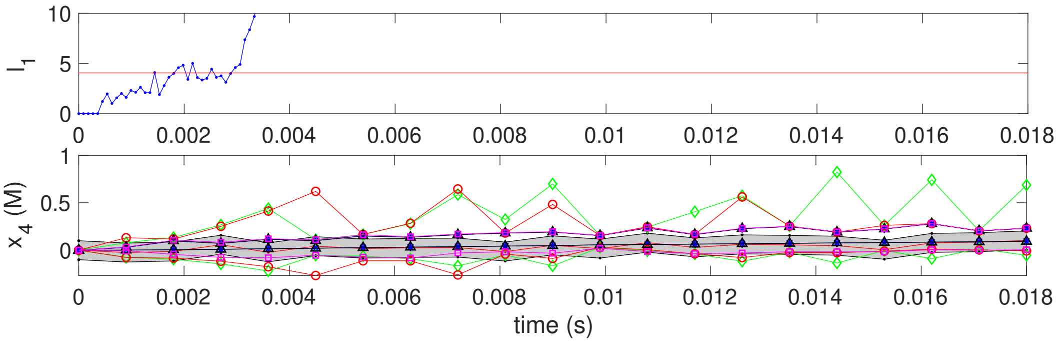

Figure 4.22: Fault detection results for Scenario (e) in Example 4.3.4 using Method (ii) (top) and Methods (iii)-(vii) (bottom). Methods (ii) declares a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds ($\star$, $\triangle$, $\square$, $\diamond$, $\circ$) have empty intersection with the bounded-error measurement (gray shaded).

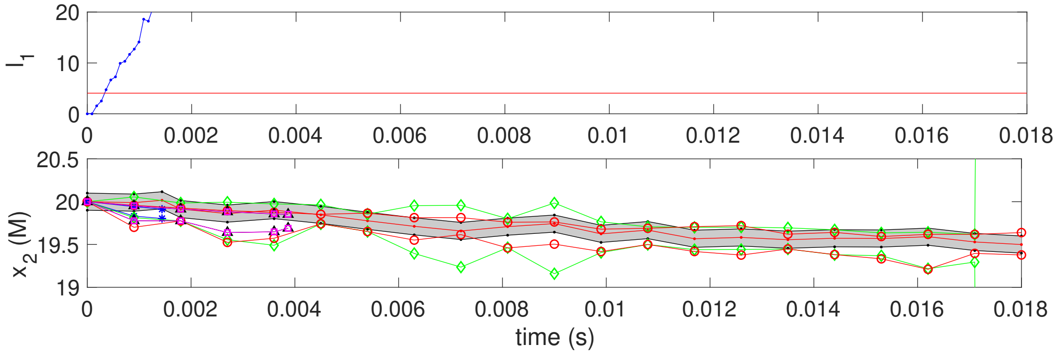

Figure 4.23: Fault detection results for Scenario (f) in Example 4.3.4 using Method (ii) (top) and Methods (iii)-(vii) (bottom). Methods (ii) declares a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds ($\star$, $\triangle$, $\square$, $\diamond$, $\circ$) have empty intersection with the bounded-error measurement (gray shaded).

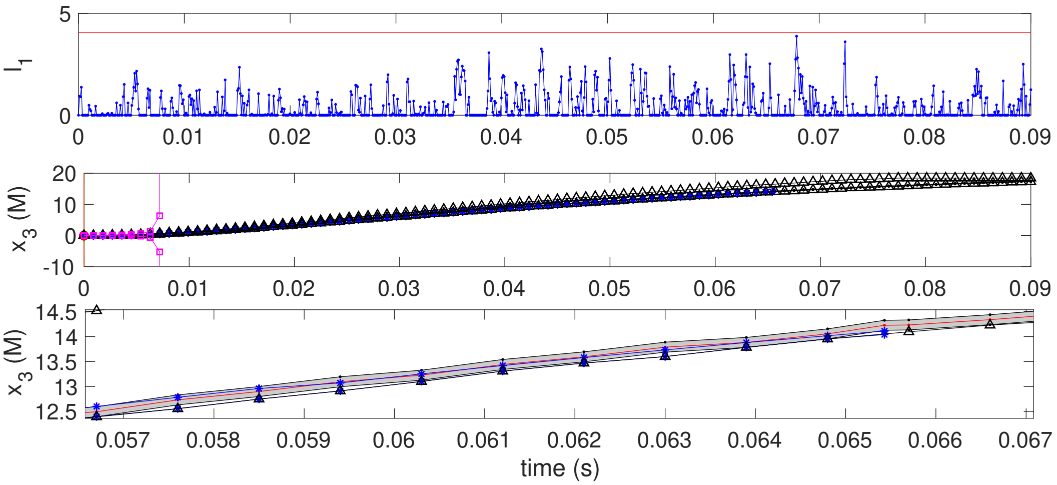

Figure 4.24: Fault detection results for Scenario (g) in Example 4.3.4 using Method (ii) (top), Methods (iii)-(vii) (middle), and zoomed-in Methods (iii)-(vii) (bottom). Methods (ii) declares a fault when their residuals (black and blue) exceed the threshold (red). Methods (iii)-(vii) declare a fault when the state estimator bounds (⋆, △, □, ◇, ○) have empty intersection with the bounded-error measurement (gray shaded).

CHAPTER 5

A COMPARISON OF ZONOTOPE ORDER REDUCTION TECHNIQUES

5.1 Introduction

Since the seminal work of Kühn [2], zonotopes have been widely adopted as an accurate and efficient way to model bounded uncertainties and noises in a variety of control applications, including reachability analysis [2, 58, 59], state estimation [60, 22, 37, 61, 4], hybrid systems verification [62, 63, 64], robust control [65, 33], and fault detection [47, 48, 66, 7]. Zonotopes are significantly more flexible than parallelotopes and ellipsoids, while requiring much less computational effort than general convex polytopes [4]. However, many operations on zonotopes yield results with higher complexity than their arguments [2], which is a serious limitation, particularly for recursive algorithms. To address this, *order reduction* methods bound a given zonotope within another of lower complexity. These methods are essential for many control algorithms, and can significantly impact their efficiency and performance. For example, inaccurate reduction can lead to overly conservative set-based estimators, and hence to conservative control actions or ineffective fault detection [64, 4].

Order reduction was first addressed in [2] in the context of reachability analysis. The first general purpose method was proposed in [60], followed shortly by a similar method in [58]. These methods (Methods 1 & 2, resp.) are both very efficient. However, while Method 1 has been overwhelmingly used in the literature [37, 61, 65, 48], there are no available studies comparing their accuracy. A more sophisticated approach was proposed in [63] (Method 3) and shown to be significantly more accurate than Method 2, but only for a limited set of tests with low-dimensional zonotopes

($n \leq 4$). Moreover, Method 1 was not compared. Unfortunately, Method 3 requires a combinatorial search that is problematic in high-dimensions (see §5.3.3). To address this, a fourth method was recently proposed in [4] (Method 4) that follows the main insights of Method 3 but eliminates the combinatorial search using an iterative matrix factorization. It was claimed in [4] that Method 4 matches the accuracy of Method 3 at significantly lower cost. However, because Method 4 was not the focus of that article, it was described only in the appendix, with no theoretical justification and no comparisons.

This Chapter makes two main contributions. First, Method 4 is presented in detail and its validity is established. Second, a comprehensive comparison of Methods 1–4 is presented considering both computational cost and overestimation error for a large test set. The effects of problem dimension, initial zonotope order, and reduced zonotope order are also investigated. The results provide valuable guidance for designing set-based estimation and control algorithms that more effectively balance accuracy with computational cost.

5.2 Preliminaries

A zonotope is a convex polytope that can be represented as a Minkowski sum of line segments, or equivalently as the image of a unit hypercube under an affine mapping [2]. Formally, $Z \subset \mathbb{R}^n$ is a zonotope iff

$$\exists (\mathbf{G}, \mathbf{c}) \in \mathbb{R}^{n \times n_g} \times \mathbb{R}^n : \; Z = \{\mathbf{G}\boldsymbol{\xi} + \mathbf{c} : \|\boldsymbol{\xi}\|_\infty \leq 1\}. \tag{5.1}$$

The vector $\mathbf{c}$ is the *center*, the n_g columns of $\mathbf{G}$ are the *generators*, and (5.1) is called the *generator-representation* (G-rep) of Z. We use the shorthand $Z = \{\mathbf{G}, \mathbf{c}\} \subset \mathbb{R}^n$ to denote zonotopes throughout. Increasing n_g makes zonotopes more flexible, but also more cumbersome. The complexity of a zonotope is described by its *order*, defined as $o \equiv n_g/n$ [58]. A first-order zonotope is a parallelotope if $\mathbf{G}$ is full rank and an interval if $\mathbf{G}$ is diagonal.

For $Z, Y \subset \mathbb{R}^n$ and $\mathbf{R} \in \mathbb{R}^{m \times n}$, define the linear mapping and Minkowsi sum, respectively, as

$$\mathbf{R}Z \equiv \{\mathbf{R}\mathbf{z} : \mathbf{z} \in Z\}, \quad Z + Y \equiv \{\mathbf{z} + \mathbf{y} : \mathbf{z} \in Z,\ \mathbf{y} \in Y\}.$$

When $Z = \{\mathbf{G}_z, \mathbf{c}_z\}$ and $Y = \{\mathbf{G}_y, \mathbf{c}_y\}$ are zonotopes, $\mathbf{R}Z$ and $Z+Y$ can be computed exactly as [2]:

$$\mathbf{R}Z = \{\mathbf{R}\mathbf{G}_z, \mathbf{R}\mathbf{c}_z\}, \quad Z + Y = \{[\mathbf{G}_z\ \mathbf{G}_y], \mathbf{c}_z + \mathbf{c}_y\}. \tag{5.2}$$

Clearly, this can be done efficiently and reliably, even in high dimensions, which is not the case for general convex polytopes [4]. However, $\mathbf{R}Z$ and $Z + Y$ can be higher order than Z and Y, and this holds for other important operations as well, such as the convex hull in [58]. This is a major drawback, particularly when such operations are applied recursively (e.g., Minkowski sums in state estimation with additive uncertainty [2, 37]).

Given $Z = \{\mathbf{G}, \mathbf{c}\} \subset \mathbb{R}^n$, *order reduction* addresses this issue by finding a lower-order zonotope Z_R that contains Z. Ideally, Z_R has minimal overestimation, which can be assessed using the following volume and Hausdorff error metrics, where $v(Z)$ is the volume of Z, $r(Z) \equiv \max_{\mathbf{z} \in Z} \|\mathbf{z} - \mathbf{c}\|_2$ is the radius of Z, and d_H is the Hausdorff distance:

$$\Theta_V \equiv \frac{v(Z_R)^{\frac{1}{n}} - v(Z)^{\frac{1}{n}}}{v(Z)^{\frac{1}{n}}}, \qquad \Theta_H \equiv \frac{d_H(Z_R, Z)}{r(Z)}. \tag{5.3}$$

Since $Z \subset Z_R$, the Hausdorff distance is given by

$$d_H(Z_R, Z) = \max_{\mathbf{y} \in Z_R} \min_{\mathbf{z} \in Z} \|\mathbf{y} - \mathbf{z}\|_2.$$

Thus, Θ_H is the maximum distance that a point in Z_R can be from Z, relative to the radius of Z, while Θ_V measures the volume added by reduction relative to the volume of Z.

Lemma 3. *The volume of* $Z = \{\mathbf{G}, \mathbf{c}\} \subset \mathbb{R}^n$ *is given by [37]:*

$$v(Z) = 2^n \sum |\det[\mathbf{g}_{s_1} \ \cdots \ \mathbf{g}_{s_n}]|,$$

where the sum runs over all combinations of n *indices* s_i *from the set* $\{1, \ldots, n_g\}$ *and* $\mathbf{g}_i$ *is the* i^{th} *column of* $\mathbf{G}$*.*

Lemma 4. *Let* $Z = \{\mathbf{G}, \mathbf{c}\}$ *and let* $Z_R = \{\mathbf{G}_R, \mathbf{c}_R\}$ *be a superset of* Z *with* $\mathbf{c}_R = \mathbf{c}$*. Then,*

$$r(Z) = \max_{\|\boldsymbol{\lambda}\|_2 = 1} \|\boldsymbol{\lambda}^{\mathrm{T}} \mathbf{G}\|_1, \tag{5.4}$$

$$d_H(Z_R, Z) = \max_{\|\boldsymbol{\lambda}\|_2 = 1} |\|\boldsymbol{\lambda}^{\mathrm{T}} \mathbf{G}_R\|_1 - \|\boldsymbol{\lambda}^{\mathrm{T}} \mathbf{G}\|_1|. \tag{5.5}$$

Proof. Define the support function $h_Z(\boldsymbol{\lambda}) \equiv \max_{\mathbf{z} \in Z} \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{z}$. It follows from a standard duality argument that $d_H(Z_R, Z) = \max_{\|\boldsymbol{\lambda}\|_2 = 1} |h_{Z_R}(\boldsymbol{\lambda}) - h_Z(\boldsymbol{\lambda})|$ (see Lemma 2 in [113]). This is equivalent to (5.5) because, by (5.1),

$$h_Z(\boldsymbol{\lambda}) = \max_{\|\boldsymbol{\xi}\|_\infty \leq 1} \boldsymbol{\lambda}^{\mathrm{T}} (\mathbf{G}\xi + \mathbf{c}) = \|\boldsymbol{\lambda}^{\mathrm{T}} \mathbf{G}\|_1 + \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{c}.$$

Moreover, (5.4) follows from (5.5) because $r(Z)$ is the Hausdorff distance between Z and the singleton $\{\mathbf{c}\}$. □ □

Lemma 5. *Let $Z = \{\mathbf{G}, \mathbf{c}\} \subset \mathbb{R}^n$, denote the elements of $\mathbf{G}$ by g_{ij}, and define $\mathbf{d} \in \mathbb{R}^n$ elementwise by $d_i \equiv \sum_j |g_{ij}|$. The interval hull of Z is given in G-rep by $\{\text{diag}(\mathbf{d}), \mathbf{c}\}$ [60].*

5.3 Order Reduction Methods

Let $Z = \{\mathbf{G}, \mathbf{c}\}$ be a zonotope with initial order $o_o = n_g/n$. To reduce Z to order $o < n_g/n$, existing methods all take the following four steps. First, the columns of $\mathbf{G}$ are reordered. It follows from (5.1) that this does not affect the set Z. Second, the reordered $\mathbf{G}$ matrix is partitioned as $[\mathbf{K}\ \mathbf{L}]$ with $\mathbf{K} \in \mathbb{R}^{n \times n(o-1)}$ and $\mathbf{L} \in \mathbb{R}^{n \times (n_g - n(o-1))}$. From (5.2), this corresponds to splitting Z into a sum of two zonotopes,

$$Z = K + L, \quad K \equiv \{\mathbf{K}, \mathbf{c}\}, \quad L \equiv \{\mathbf{L}, \mathbf{0}\}.$$

Third, L is overapproximated by a first order zonotope $L_R \equiv \{\mathbf{L}_R, \mathbf{0}\}$ with $\mathbf{L}_R \in \mathbb{R}^{n \times n}$. Finally, Z is overapproximated by

$$Z_R \equiv K + L_R = \{[\mathbf{K}\ \mathbf{L}_R], \mathbf{c}\}. \tag{5.6}$$

It is readily verified that this eliminates all but $n \times o$ generators, as desired. Methods 1–4 are now described in detail.

5.3.1 Method 1

Method 1 [60] chooses L_R as the interval hull of L, which is easily computed as in Lemma 5. Clearly, it is desirable to choose L so that the overestimation introduced by taking its interval hull is minimized. Method 1 aims to achieve this by choosing $\mathbf{L}$ as the $n_g - n(o-1)$ shortest generators in $\mathbf{G}$. This is implemented in Algorithm 4, where the subroutine IntervalHull($\mathbf{L}$) returns the generator matrix of the interval hull of L. The complexity of Algorithm 4 is dominated by line 2, with $\mathcal{O}(nn_g)$ for computing two-norms and $\mathcal{O}(n_g \log n_g)$ for sorting, for a total of $\mathcal{O}(n_g(n + \log n_g))$, or $\mathcal{O}(no_0(n + \log(no_0)))$.

Algorithm 4 Reduces $\{\mathbf{G}, \mathbf{c}\}$ to order o using Method 1

```
1: procedure ReduceOrder1($\mathbf{G}$,$n$,$n_g$,$o$)
2:     Reorder the columns of $\mathbf{G}$ by decreasing two-norm
3:     $\mathbf{K} \leftarrow \mathbf{G}_{1:n,1:n(o-1)}$
4:     $\mathbf{L} \leftarrow \mathbf{G}_{1:n,n(o-1)+1:n_g}$
5:     $\mathbf{L}_R \leftarrow$ IntervalHull($\mathbf{L}$)
6:     return $\begin{bmatrix}\mathbf{K} & \mathbf{L}_R\end{bmatrix}$
7: end procedure
```

5.3.2 Method 2

Method 2 [58] also chooses L_R as the interval hull of L, but aims to minimize the error by making L interval-shaped. Specifically, $\mathbf{L}$ is chosen as the $n_g - n(o-1)$ generators $\mathbf{g}_j$ that have the smallest values of the score

$$\gamma_j \equiv \|\mathbf{g}_j\|_1 - \|\mathbf{g}_j\|_\infty,$$

which measures how nearly axis-aligned $\mathbf{g}_j$ is and is zero when $\mathbf{g}_j$ is a scaled unit vector. This is implemented exactly as in Algorithm 4 by simply replacing line 2. The complexity is again $\mathcal{O}(n_g(n + \log n_g)) = \mathcal{O}(no_0(n + \log(no_0)))$.

5.3.3 Method 3

Method 3 was proposed in [63, 64] to reduce the conservatism of Methods 1 and 2 using the key new idea of enclosing L with a parallelotope rather than an interval. As in Method 1, $\mathbf{L}$ is chosen as the $n_g - n(o-1)$ shortest generators in $\mathbf{G}$. Next, an invertible matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$ is chosen that defines a parallelotope $T \equiv \{\mathbf{T}, \mathbf{0}\}$, and L_R is chosen as the minimum volume parallelotope with the same 'shape' as T that encloses L. Precisely, L_R is chosen from the family of parallelotopes $T_{\mathbf{D}} \equiv \{\mathbf{TD}, \mathbf{0}\}$, where $\mathbf{D} \in \mathbb{R}^{n\times n}$ is a diagonal scaling matrix. The smallest such set enclosing L is [63]:

$$L_R \equiv \{\mathbf{T} \times \textsc{IntervalHull}(\mathbf{T}^{-1}\mathbf{L}), \mathbf{0}\}. \tag{5.7}$$

In [64], $\mathbf{T}$ is chosen as a combination of n generators in $\mathbf{L}$. The choice that minimizes $v(L_R)$ is desirable since this also minimizes the volume error $v(L_R) - v(L)$ (although not necessarily $v(Z_R) - v(Z)$ via (5.6)). However, computing $v(L_R)$ is $\mathcal{O}(n^3)$ and there are a huge number of combinations $\mathbf{T}$ (i.e., $\binom{n_g - o(n-1)}{n}$). Thus, we implemented a more practical heuristic that is closest to Method C in [64] and follows the implementation in the code CORA [114]. It requires two parameters $\kappa_1, \kappa_2 \leq n_g$ and is described in Algorithm 5. First, κ_1 candidate generators are selected from $\mathbf{L}$ based on their two-norm. Second, candidate $\mathbf{T}$ matrices are generated from all combinations of n generators from this restricted group, and these are ordered in terms of the volume of the corresponding parallelotope $\{\mathbf{T}, \mathbf{0}\}$. From this, the κ_2 choices of $\mathbf{T}$ with largest volume are selected. A key idea here is that, if $\{\mathbf{T}, \mathbf{0}\}$ (which is a subset of L) has large volume, then it should not need to be enlarged by much in order to enclose L, which will tend to minimize the over-approximation error. Next, an enclosure L_R is computed as in (5.7) for each of the κ_2 choices of $\mathbf{T}$, and the enclosure with minimum volume is selected. The complexity of Algorithm 5 is $\mathcal{O}(no_0(n+\log(no_0)) + \binom{\kappa_1}{n}n^3 + \kappa_2 n^3(o_0 - o))$, where the three terms correspond to, respectively, the initial sort in line 2, the loop in lines 9–12, and the loop in lines 14–18. In all of the comparisons in this study, we use the heuristics $\kappa_1 = n + 8$ and $\kappa_2 = n + 3$ suggested in [64].

5.3.4 Method 4

Method 4 was introduced in [4] to achieve accurate reductions using the main ideas of Method 3 with lower computational cost. Like Method 3, Method 4 also chooses a

Algorithm 5 Reduces $\{\mathbf{G}, \mathbf{c}\}$ to order o using Method 3

```
procedure ReduceOrder3($\mathbf{G}$,$n$,$n_g$,$o$,$\kappa_1$,$\kappa_2$)
    Reorder the columns of $\mathbf{G}$ by decreasing two-norm
    $\mathbf{K} \leftarrow \mathbf{G}_{1:n,1:n(o-1)}$
    $\mathbf{L} \leftarrow \mathbf{G}_{1:n,n(o-1)+1:n_g}$
    $\mathbf{L}' \leftarrow [\text{IntervalHull}(\mathbf{L})]^{-1}\mathbf{L}$        ▷ Scale $\mathbf{L}$
    Reorder the columns of $\mathbf{L}'$ by decreasing two-norm
    $\mathbf{L}'' \leftarrow \mathbf{L}'_{1:n,1:\kappa_1}$
    $\mathcal{L} \leftarrow$ collection of all combinations of $n$ columns from $\mathbf{L}''$
    for $l = 1$ to $|\mathcal{L}|$ do
        $\mathbf{T} \leftarrow \mathcal{L}(l)$
        $v_l \leftarrow \text{Volume}(\mathbf{T})$
    end for
    $E \leftarrow$ set of indices $l$ of the $\kappa_2$ largest values of $v_l$
    for $l \in E$ do
        $\mathbf{T} \leftarrow \mathcal{L}(l)$
        $\mathbf{L}_R \leftarrow \mathbf{T} \times \text{IntervalHull}(\mathbf{T}^{-1}\mathbf{L})$
        $v_l^* \leftarrow \text{Volume}(\mathbf{L}_R)$
    end for
    $l \leftarrow \text{argmin}_{l \in E} v_l^*$
    $\mathbf{T} \leftarrow \mathcal{L}(l)$
    $\mathbf{L}_R \leftarrow \mathbf{T} \times \text{IntervalHull}(\mathbf{T}^{-1}\mathbf{L})$
    return $\begin{bmatrix}\mathbf{K} & \mathbf{L}_R\end{bmatrix}$
end procedure
```

matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$ defining a parallelotope that is used to enclose L. However, $\mathbf{T}$ is specified as a selection of n generators from $\mathbf{G}$, rather than from $\mathbf{L}$, and the partitioning of $\mathbf{G}$ into $[\mathbf{K}\ \mathbf{L}]$ is decided after $\mathbf{T}$ is determined using a novel volume-error heuristic. The first step is to reorder the columns of $\mathbf{G}$ to obtain $[\mathbf{T}\ \mathbf{V}]$. As in Method 3, the aim is to find $\mathbf{T}$ such that $\{\mathbf{T}, \mathbf{0}\}$ has large volume. Method 4 accomplishes this using a greedy matrix factorization algorithm (Algorithm 6) based on the following lemma.

Lemma 6. *Let* $\mathbf{T} \in \mathbb{R}^{n \times n}$ *be invertible and let* $T \equiv \{\mathbf{T}, \mathbf{0}\}$*. Choose* $\mathbf{v} \in \mathbb{R}^n$ *and, for each* i*, let* T^i *be the parallelotope constructed by replacing the* i^{th} *column of* $\mathbf{T}$ *by* $\mathbf{v}$*. Then* $v(T^i) = |r_i| v(T)$ *for all* $i \in \{1, \ldots, n\}$*, where* $\mathbf{r} \equiv \mathbf{T}^{-1}\mathbf{v}$*.*

Proof. Since $\mathbf{T}^{-1}\mathbf{v} = \mathbf{r}$, we have

$$\mathbf{T}^{-1}\mathbf{T}^i = \begin{bmatrix}\mathbf{e}_1 & \cdots & \mathbf{e}_{i-1} & \mathbf{r} & \mathbf{e}_{i+1} & \cdots & \mathbf{e}_n\end{bmatrix}.$$

Noting that $\mathbf{r} = \sum_{i=1}^{n} r_i \mathbf{e}_i$ and using standard properties of the determinant gives

$$\det(\mathbf{T}^{-1}\mathbf{T}^i) = \det \begin{bmatrix} \mathbf{e}_1 & \cdots & \mathbf{e}_{i-1} & r_i\mathbf{e}_i & \mathbf{e}_{i+1} & \cdots & \mathbf{e}_n \end{bmatrix} = r_i.$$

Thus, $|r_i| = |\det(\mathbf{T}^{-1}\mathbf{T}^i)| = \frac{|\det(\mathbf{T}^i)|}{|\det(\mathbf{T})|}$, which equals $\frac{v(T^i)}{v(T)}$ by Lemma 3. □ □

Algorithm 6 transforms $\mathbf{G}$ to $[\mathbf{T}\ \mathbf{V}]$ by iteratively swapping columns into $\mathbf{T}$ that increase $v(\{\mathbf{T}, \mathbf{0}\})$ according to Lemma 6. A similar algorithm is used in [115, 116] in the context of low-rank matrix approximation. Given $\delta \geq 0$, Algorithm 6 terminates with $\mathbf{T}$ such that $v(\{\mathbf{T}, \mathbf{0}\})$ cannot be increased by more than a factor of $(1+\delta)$ by any *single* column swap, but it does not ensure that $v(\{\mathbf{T}, \mathbf{0}\})$ is maximal. Choosing $\delta > 0$ reduces the complexity, as shown below. Notably, Algorithm 6 does not require any volume computations.

Theorem 7. *Choose any* $\mathbf{G} \in \mathbb{R}^{n \times n_g}$ *and any* $\epsilon, \delta \geq 0$. *Let* $L \in \mathbb{R}_+$ *satisfy* $L \geq \|\mathbf{g}_i\|_2$ *for every column* $\mathbf{g}_i$ *of* $\mathbf{G}$. *If* $\mathbf{G}$ *is full rank and all pivot elements selected in line 7 of Algorithm 6 satisfy* $|g^*_{ij}| > \epsilon$, *then* FACTORG$(\mathbf{G}, \epsilon, \delta)$ *terminates finitely after* M *passes through the while loop on lines 16–21, where*

$$M \leq \min\left(n \frac{\log(L/\epsilon)}{\log(1+\delta)}, \binom{n_g}{n} \right). \tag{5.8}$$

Moreover, upon termination (a) $\mathbf{T} \equiv \mathbf{G}_{1:n,1:n}$ *is invertible, (b)* $\mathbf{G}^* = \mathbf{T}^{-1}\mathbf{G}$, *and (c)* $|g^*_{ij}| \leq 1 + \delta$ *for all* i *and* j.

Proof. Let $\mathbf{G}_0$ and $\mathbf{G}^*_0$ denote the matrices stored as $\mathbf{G}$ and $\mathbf{G}^*$ when line 15 is reached for the first time. Since lines 5–13 bring $\mathbf{G}$ to reduced row echelon form, $\mathbf{G}^*_0 = [\mathbf{I}_{n\times n}\ \mathbf{R}_0]$ for some $\mathbf{R}_0 \in \mathbb{R}^{n\times(n_g-n)}$. Line 10 mimics all column swaps on the original $\mathbf{G}$ matrix. Thus, with $\mathbf{G}_0 = [\mathbf{T}_0\ \mathbf{V}_0]$, it follows that $[\mathbf{T}_0\ \mathbf{V}_0]$ can be transformed to $[\mathbf{I}_{n\times n}\ \mathbf{R}_0]$ by elementary row operations. Therefore, $\mathbf{T}_0$ is invertible and $\mathbf{G}^*_0 = \mathbf{T}_0^{-1}\mathbf{G}_0$.

Algorithm 6 Reorders columns of $\mathbf{G}$ by mimicking column swaps needed to transform $\mathbf{G}$ to a reduced row echelon form $\mathbf{G}^*$ with all $|g^*_{ij}| \le 1 + \delta$.

```
 1: procedure FactorG($\mathbf{G}, \epsilon, \delta$)
 2:     $\mathbf{G}^* \leftarrow \mathbf{G}$
 3:     ▷ This loop brings $\mathbf{G}^*$ to reduced row echelon form and
 4:     ▷ mimics all column swaps on $\mathbf{G}$
 5:     for $k = 1$ to $n$ do
 6:         Normalize rows $k$ through $n$ of $\mathbf{G}^*$ by their 1-norms
 7:         $(i, j) \leftarrow \underset{i \in \{k,\dots,n\},\ j \in \{k,\dots,n_g\}}{\operatorname{argmax}} |g^*_{ij}|$
 8:         If $|g^*_{ij}| \le \epsilon$, break loop
 9:         Swap rows $k$ and $i$ of $\mathbf{G}^*$
10:         Swap columns $k$ and $j$ of $\mathbf{G}$
11:         Swap columns $k$ and $j$ of $\mathbf{G}^*$
12:         Transform the $k^{\text{th}}$ column of $\mathbf{G}^*$ to $\mathbf{e}_k$ by row operations
13:     end for
14:     ▷ Do extra column swaps until all $|g^*_{ij}| \le 1 + \delta$
15:     $(k, j) \leftarrow \underset{k \in \{1,\dots,n\},\ j \in \{1,\dots,n_g\}}{\operatorname{argmax}} |g^*_{kj}|$
16:     while $|g^*_{kj}| > 1 + \delta$ do                    ▷ Entering this loop implies $j > n$
17:         Swap columns $k$ and $j$ of $\mathbf{G}$
18:         Swap columns $k$ and $j$ of $\mathbf{G}^*$
19:         Transform the $k^{\text{th}}$ column of $\mathbf{G}^*$ to $\mathbf{e}_k$ by row operations
20:         $(k, j) \leftarrow \underset{k \in \{1,\dots,n\},\ j \in \{1,\dots,n_g\}}{\operatorname{argmax}} |g^*_{kj}|$
21:     end while
22:     return $\mathbf{G}$, $\mathbf{G}^*$
23: end procedure
```

To set up an inductive argument, suppose that line 16 is reached after $m \ge 0$ passes through the while loop (lines 16–21) with $\mathbf{G}_m = [\mathbf{T}_m\ \mathbf{V}_m]$, $\mathbf{G}^*_m = [\mathbf{I}\ \mathbf{R}_m]$, and $\mathbf{G}^*_m = \mathbf{T}_m^{-1}\mathbf{G}_m$. Denote $v_m = v(\{\mathbf{T}_m, \mathbf{0}\})$ and suppose that $|g^*_{kj}| > 1+\delta$. Then, noting that $j > n$, let $\mathbf{v}$ be the j^{th} column of $\mathbf{G}_m$ and let $\mathbf{r} = \mathbf{g}^*_j = \mathbf{T}_m^{-1}\mathbf{v}$. Examining the code inside the while loop, $\mathbf{G}_{m+1} = [\mathbf{T}_{m+1}\ \mathbf{V}_{m+1}]$ is formed by swapping the k^{th} column of $\mathbf{T}_m$ with $\mathbf{v}$, while $\mathbf{G}^*_{m+1} = [\mathbf{I}\ \mathbf{R}_{m+1}]$ is formed by swapping the k^{th} column of $\mathbf{I}$ with $\mathbf{r}$ and then recovering reduced row echelon form by elementary row operations. Since identical column swaps are done on $\mathbf{G}_m$ and $\mathbf{G}^*_m$, $\mathbf{G}_{m+1}$ must again be reducible to $\mathbf{G}^*_{m+1}$ by elementary row operations, which implies that $\mathbf{T}_{m+1}$ is invertible and $\mathbf{G}^*_{m+1} = \mathbf{T}_{m+1}^{-1}\mathbf{G}_{m+1}$. Moreover, Lemma 6 gives $v_{m+1} = |r_k|v_m = |g^*_{kj}|v_m > (1+\delta)v_m$. Thus, $\{v_m\}$ is strictly increasing and $\{\mathbf{T}_m\}$ has no repeats. Since there are only $\binom{n_g}{n}$

choices of $\mathbf{T}$, the algorithm terminates after $M \leq \binom{n_g}{n}$ iterations. Conclusions (a) and (b) follow by induction, and (c) follows from line 16.

To improve the $M \leq \binom{n_g}{n}$ bound when $\epsilon, \delta > 0$, note that Hadamard's inequality implies $v_m = v(\{\mathbf{T}_m, \mathbf{0}\}) = 2^n|\det(\mathbf{T}_m)| \leq 2^n L^n$ for all m. On the other hand, $v_0 = 2^n|\det(\mathbf{T}_0)| > 2^n\epsilon^n$ because $|\det(\mathbf{T}_0)|$ is the product of the pivot values $|g^*_{ij}| > \epsilon$ selected in line 7. Since $v_{m+1} > (1+\delta)v_m$, M must satisfy $2^n\epsilon^n(1+\delta)^M \leq v_M \leq 2^n L^n$, which yields (5.8). □ □

For reasonable $L, \epsilon, \delta > 0$, the term $n\frac{\log(L/\epsilon)}{\log(1+\delta)}$ is much less than $\binom{n_g}{n}$. Moreover, computational experience shows that Algorithm 6 typically requires dramatically fewer than $n\frac{\log(L/\epsilon)}{\log(1+\delta)}$ iterations because v_m increases by more than the minimum factor $(1+\delta)$ in each iteration. For example, using 100 randomly generated zonotopes with order 10, dimension 100, and $\|\mathbf{g}_i\|_2 \leq L = 60$, the average number of iterations with $\epsilon = 10^{-6}$ and $\delta = 10^{-3}$ was a mere 15.37, compared to $n\frac{\log(L/\epsilon)}{\log(1+\delta)} \geq 10^6$ and $\binom{n_g}{n} \geq 10^{100}$.

After $\mathbf{T}$ is computed, Method 4 chooses $\mathbf{L}$ to be composed of the generators in $\mathbf{T}$ (which are never eliminated) and a selection of other generators from $\mathbf{G}$ that are chosen and eliminated one at a time, as described in Algorithm 7. In each iteration, the next generator to be eliminated is chosen based on the following result.

Lemma 7. *Let $Z = \{\mathbf{G}, \mathbf{c}\} \subset \mathbb{R}^n$, let $\mathbf{T} = \mathbf{G}_{1:n,1:n}$ be invertible, and define $\mathbf{G}^* \equiv \mathbf{T}^{-1}\mathbf{G}$. Choose $j > n$ and let $\mathbf{v}$ and $\mathbf{r}$ denote the j^{th} columns of $\mathbf{G}$ and $\mathbf{G}^*$, respectively. The order $\frac{n+1}{n}$ zonotope $X \equiv \{[\mathbf{T}\ \mathbf{v}], \mathbf{c}\}$ is enclosed by the parallelotope $X_R \equiv \{\mathbf{T}(\mathbf{I} + \operatorname{diag}|\mathbf{r}|), \mathbf{c}\}$, and*

$$\frac{v(X_R) - v(X)}{v(\{\mathbf{T}, \mathbf{c}\})} = \Pi_{i=1}^n (1 + |r_i|) - (1 + \|\mathbf{r}\|_1). \tag{5.9}$$

Moreover, if $\|\mathbf{r}\|_\infty \leq 1$, then X_R is the minimum volume parallelotope enclosing X.

Proof. Theorem 3 in [13] proves that X_R encloses X and is minimal when $\|\mathbf{r}\|_\infty \leq 1$.

By Lemma 3,

$$v(X_R) = 2^n|\det(\mathbf{T}(\mathbf{I} + \text{diag}|\mathbf{r}|))| = 2^n|\det \mathbf{T}|\Pi_{i=1}^n(1 + |r_i|).$$

Moreover, letting $\mathbf{t}_i$ denote the i^{th} column of $\mathbf{T}$,

$$\begin{aligned} v(X) &= 2^n(|\det \mathbf{T}| + \textstyle\sum_{i=1}^n|\det[\mathbf{t}_1 \cdots \mathbf{t}_{i-1} \ \mathbf{v} \ \mathbf{t}_{i+1} \cdots \mathbf{t}_n]|), \\ &= 2^n(|\det \mathbf{T}| + \textstyle\sum_{i=1}^n|\det(\mathbf{T}[\mathbf{e}_1 \cdots \mathbf{e}_{i-1} \ \mathbf{r} \ \mathbf{e}_{i+1} \cdots \mathbf{e}_n])|), \\ &= 2^n|\det \mathbf{T}|(1 + \textstyle\sum_{i=1}^n|r_i|). \end{aligned}$$

Thus, (5.9) follows using $v(\{\mathbf{T}, \mathbf{c}\}) = 2^n|\det \mathbf{T}|$. □ □

Given $\mathbf{G} = [\mathbf{T} \ \mathbf{V}]$ and $\mathbf{G}^* = \mathbf{T}^{-1}\mathbf{G} = [\mathbf{I} \ \mathbf{R}]$ with all $|r_{ij}| \leq 1 + \delta$, Method 4 proceeds by first selecting the column $\mathbf{r}$ of $\mathbf{R}$ that minimizes the error (5.9). The corresponding column $\mathbf{v}$ of $\mathbf{V}$ is then removed to form $\mathbf{V}_-$ and eliminated as follows:

$$\begin{aligned} Z = \{[\mathbf{T} \ \mathbf{V}], \mathbf{c}\} &= \{\mathbf{V}_-, \mathbf{c}\} + \{[\mathbf{T} \ \mathbf{v}], \mathbf{0}\} \\ &\subset \{\mathbf{V}_-, \mathbf{c}\} + \{\mathbf{T}(\mathbf{I} + \text{diag}|\mathbf{r}|), \mathbf{0}\} \\ &= \{[\mathbf{T}(\mathbf{I} + \text{diag}|\mathbf{r}|) \ \mathbf{V}_-], \mathbf{c}\}. \end{aligned}$$

When $\delta > 0$, this reduction of $\{[\mathbf{T} \ \mathbf{v}], \mathbf{0}\}$ may not be optimal because $\|\mathbf{r}\|_\infty$ may not be less than 1. However, we expect it to be nearly optimal for small δ. This strategy is then repeated in the next iteration following the updates $\mathbf{V}' = \mathbf{V}_-$, $\mathbf{T}' = \mathbf{T}(\mathbf{I}+\text{diag}|\mathbf{r}|)$, and $\mathbf{R}' = (\mathbf{T}')^{-1}\mathbf{V}' = [(\mathbf{I} + \text{diag}|\mathbf{r}|)]^{-1}\mathbf{T}^{-1}\mathbf{V}_- = [(\mathbf{I} + \text{diag}|\mathbf{r}|)]^{-1}\mathbf{R}_-$, where $\mathbf{R}_-$ is obtained by simply removing column $\mathbf{r}$ from $\mathbf{R}$. Note that $\mathbf{R}'$ retains the property that all elements are less than $1 + \delta$ in magnitude. We have proven the following:

Theorem 8. *Let* $Z = \{\mathbf{G}, \mathbf{c}\} \subset \mathbb{R}^n$ *and assign* $\mathbf{G}_R \leftarrow \text{ReduceOrder4}(\mathbf{G}, n, n_g, o, \epsilon, \delta)$. *Then* $Z \subset Z_R \equiv \{\mathbf{G}_R, \mathbf{c}\}$ *and* Z_R *has order* o.

Algorithm 7 Reduces $\{\mathbf{G}, \mathbf{c}\}$ to order o using Method 4

```
 1: procedure ReduceOrder4($\mathbf{G}$,$n$,$n_g$,$o$,$\epsilon$,$\delta$)
 2:     $(\mathbf{G}, \mathbf{G}^*) \leftarrow$ FactorG$(\mathbf{G}, \epsilon, \delta)$
 3:     while $n_g/n > o$ do
 4:         $j \leftarrow \underset{j=n+1,\ldots,n_g}{\operatorname{argmin}} \left[\Pi_i(1+|g^*_{ij}|) - (1+\sum_i |g^*_{ij}|)\right]$
 5:         $\mathbf{r} \leftarrow j^{\text{th}}$ column of $\mathbf{G}^*$
 6:         Remove column $j$ from $\mathbf{G}$ and $\mathbf{G}^*$
 7:         $n_g \leftarrow n_g - 1$
 8:         $\mathbf{G}_{1:n,1:n} \leftarrow \mathbf{G}_{1:n,1:n}[\operatorname{diag}(\mathbf{1} + |\mathbf{r}|)]$
 9:         $\mathbf{G}^*_{1:n,n+1:n_g} = [\operatorname{diag}(\mathbf{1} + |\mathbf{r}|)]^{-1}\mathbf{G}^*_{1:n,n+1:n_g}$
10:     end while
11:     return $\mathbf{G}$
12: end procedure
```

For fixed $\epsilon, \delta > 0$ and zonotopes with bounded generators $\|\mathbf{g}_i\|_2 \leq L$, Algorithm 7 has worst-case complexity $\mathcal{O}(n\frac{\log(L/\epsilon)}{\log(1+\delta)}n^2 o_0 + n(o_0 - o)n^2 o_0) = \mathcal{O}(n^3(o_0 - o)o_0)$, where the first term is the complexity of Algorithm 6 and the second results from the $n(o_0 - o)$ passes through the while loop on lines 3–10.

5.4 Numerical Comparisons

This section compares Methods 1–4 on several reduction tasks. All results are averaged over 500 zonotopes $Z = \{\mathbf{G}, \mathbf{c}\}$ generated by sampling $[\mathbf{G}\ \mathbf{c}]$ elementwise from a uniform distribution on $[-1, 1]$ and then scaling each column $\mathbf{g}_i$ (and $\mathbf{c}$) by $\alpha_i/\|\mathbf{g}_i\|_2$, where α_i is a uniform random scalar in $[0, 60]$. Due to the excessive cost of the combinatorial procedures in Lemma 3 and Algorithm 5, volume errors Θ_V (see (5.3)) are not shown for $n > 8$, and Method 3 is not compared for $n > 10$. Moreover, Hausdorff distances and radii are approximated using (5.4)–(5.5) by maximizing over 500 random $\boldsymbol{\lambda}$s with $\|\boldsymbol{\lambda}\|_2 = 1$. These approximations are sharp in the sense that they change by $< 1\%$ when using 2000 random $\boldsymbol{\lambda}$s for random zonotopes up to $n = 100$. Finally, we report wall-clock times for MATLAB R2015a on a Dell Precision T1700 with an i5-4690 CPU @ 3.50GHz and 16.0 GB RAM.

5.4.1 Reducing zonotopes by a single order

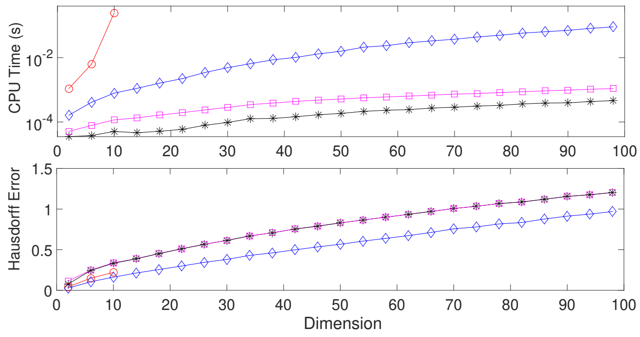

Figure 5.1: Average time and Hausdorff error Θ_H for reducing a zonotope from order 5 to 4 using Methods 1–4 ($\square$,$\star$,$\circ$,$\diamond$, resp.).

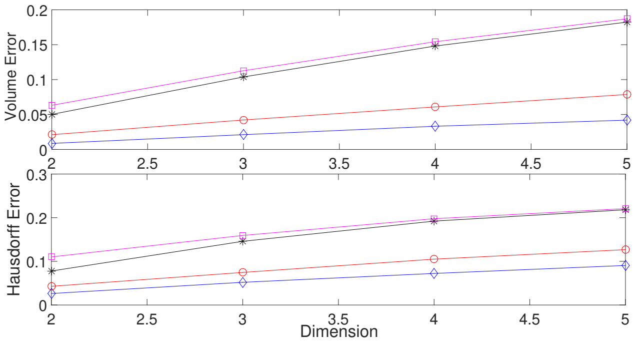

Figure 5.2: Average volume and Hausdorff errors for reducing a zonotope from order 5 to 4 using Methods 1–4 ($\square$,$\star$,$\circ$,$\diamond$, resp.).

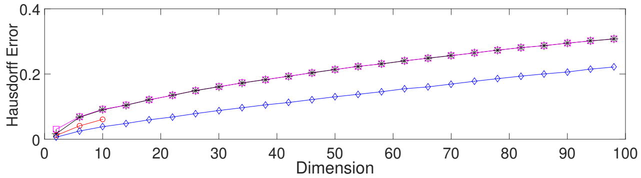

Figure 5.3: Average Hausdorff error Θ_H for reducing a zonotope from order 10 to 9 using Methods 1–4 ($\square$,$\star$,$\circ$,$\diamond$, resp.).

Figures 5.1–5.3 compare Methods 1–4 for single-order reductions with variable dimension n. Method 3 is more accurate than Methods 1–2 for $n \leq 10$, but its cost rapidly increases with n. Method 4 is slightly more accurate than Method 3, but with much lower cost, enabling accurate reduction up to $n = 98$. Clearly, Method 4 follows the empirical $\mathcal{O}(n^3)$ complexity discussed in §5.3.4 rather than the exponential worst-case estimate. For small n, Figure 5.2 shows that Θ_V and Θ_H have qualitatively similar trends.

Figure 5.3 shows Θ_H trends similar to Figure 5.1 for initial order 10 instead of 5. Time trends (not shown) are also similar, with Method 3 reaching 0.2466s at $n = 10$ and Method 4 reaching 0.1782s at $n = 98$. Finally, Figure 5.4 shows the effect of initial order with $n = 4$. Interestingly, the reduction error is greatly reduced with increasing order, while computation times are nearly constant at 1.1×10^{-4}s, 4.9×10^{-5}s, 2.1×10^{-3}s, and 3.3×10^{-4}s for Methods 1–4, respectively.

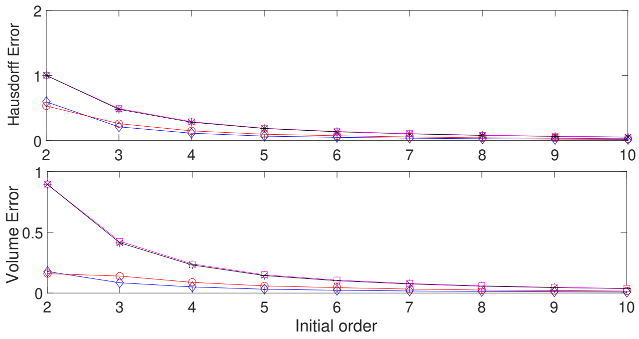

Figure 5.4: Average Hausdorff and volume errors for reducing zonotopes with dimension $n = 4$ by one order from various initial orders using Methods 1–4 (□,⋆,○,◇, resp.).

5.4.2 Reducing zonotopes by multiple orders

Figures 5.5–5.6 compare Methods 1–4 for reducing zonotopes by three orders at a time. All methods achieve this by a single call to Algorithm 4, 5, or 7 with input $o = o_0 - 3$. A variant of Method 1 is also shown (▽) that calls Algorithm 4 three times, reducing one order each time. The results differ due to the column ordering on line 2. We found this distinction unclear in [60], although the single call is markedly more accurate. Method 2 gives identical results with one or three calls, while sequential calls to Methods 3–4 were not considered because this significantly increases their complexity.

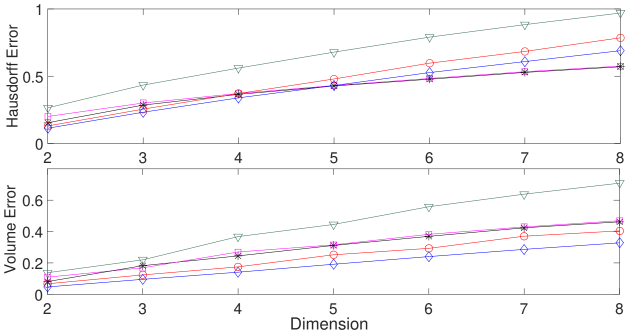

Figure 5.5: Average Hausdorff and volume errors for reducing zonotopes from order 6 to 3 using Methods 1–4 (□,⋆,○,◇, resp.) and a variant of Method 1 using 3 sequential calls to Algorithm 4 (▽).

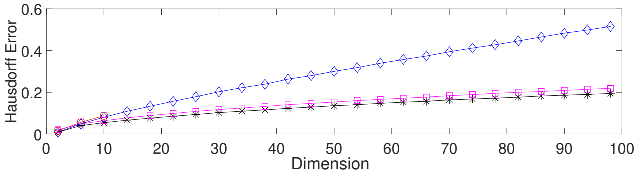

Figure 5.6: Average Hausdorff error for reducing zonotopes from order 20 to 17 using Methods 1–4 (□,⋆,○,◇, resp.).

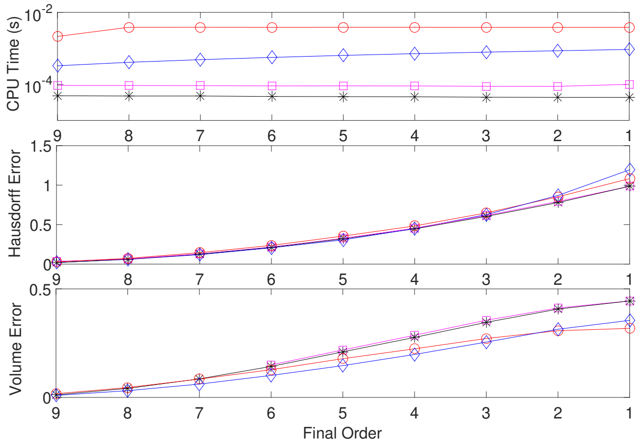

Figure 5.7: Average time and errors for reducing zonotopes with $n = 4$ from $o_0 = 10$ to different final orders using Methods 1–4 (□,⋆,○,◇).

Volume errors in Figure 5.5 are similar but larger than in Figure 5.2, with Methods 3–4 more accurate than 1–2. Surprisingly, however, Θ_H favors Methods 1–2 over 3–4 when $n > 5$ (see also Fig. 5.6). Thus, Methods 1–2 may be desirable for multi-order reductions with large n, although its likely that Θ_V would continue to favor Methods 3–4 for large n. When reducing by many orders for $n = 4$, Figure 5.7 shows that times increase modestly, Θ_H increases significantly but similarly for all methods, and Θ_V again favors Methods 3–4.

5.4.3 Reducing zonotopes in reachability analysis

Figure 5.8 compares methods for reducing zonotopes enclosing the reachable sets of random discrete-time linear systems $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{w}_k$ generated by the MATLAB routine `drss` with $n_x = n_w = 4$. Each system has bounded initial conditions and disturbances lying in random first-order zonotopes X_0 and W. The reachable sets are defined recursively by

$$R_0 = X_0, \quad R_{k+1} \equiv \{\mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{w}_k : (\mathbf{x}_k, \mathbf{w}_k) \in R_k \times W\}.$$

It is well known that each R_k is a zonotope [2]. However, the order of R_k increases linearly with k. Thus, it is common to consider low-order enclosures computed recursively by

$$\hat{X}_0 = X_0, \quad \hat{X}_{k+1} = \mathrm{Red}[\mathbf{A}\hat{X}_k + \mathbf{B}W], \tag{5.10}$$

where the set operations are computed as in (5.2) and $\mathrm{Red}[Z]$ reduces Z to a desired order o. Figure 5.8 compares Methods 1–4 for this reduction with $o = 8$. In addition, Figure 5.8 also compares variants denoted as Methods 1'–4'. Method i' works by first removing the n largest two-norm generators from Z, then reducing the remaining zonotope to order 7 using Method i, and finally replacing the n removed generators.

This simple modification is designed to mitigate unstable growth of the enclosures over long horizons based on the key insights in [2]. Note that this change has no effect on Methods 1 and 3, since they already use an initial generator sorting based on two-norms (see Algorithms 4 and 5).

Figure 5.8 shows that Methods 1–2 are the most efficient, with Method 4 about 10× slower and Method 3 about 100× slower. The Hausdorff and volume errors for Methods 1–2 grow roughly linearly in k. Method 3 is comparable to 1–2 in Θ_H, but much more accurate in Θ_V. Over short horizons, Method 4 is more accurate than Methods 1–3 in both metrics ($k \leq 30$ for Θ_H and $k \leq 50$ for Θ_V). However, Method 4 is unstable over longer horizons. Experiments show that Method 4 performs very well for the majority of random systems, but exhibits dramatic exponential growth for a small number of systems, leading to poor performance on average (e.g., Θ_H exceeded 30 at $k = 100$ for 852 of 10^4 systems, with some cases exceeding 10^6, while no other method exceeds 30 even once). This instability is not fully understood at present. However, Method 4' largely corrects the problem, providing the tightest enclosure in both Θ_H and Θ_V out to $k = 100$. In contrast, the performance of Methods 1'–3' is nearly identical to that of Methods 1–3.

We also compared the accuracy of Method 4' with $o = 8$ to Methods 1–2 with $o = 60$, where 60 was chosen to make the CPU times for the three methods roughly equal at 3.3×10^{-4}s per time step. In this case, Methods 1–2 provide the exact reachable set for $k \leq 60$ since no reduction is necessary. Despite this, Method 4' becomes the best in terms of Θ_V at $k = 77$, and stabilizes with Θ_V about 10% smaller than that of Methods 1–2 (not shown). However, Method 4' is not competitive in terms of Θ_H due to exponential growth for some random systems prior to $k = 150$.

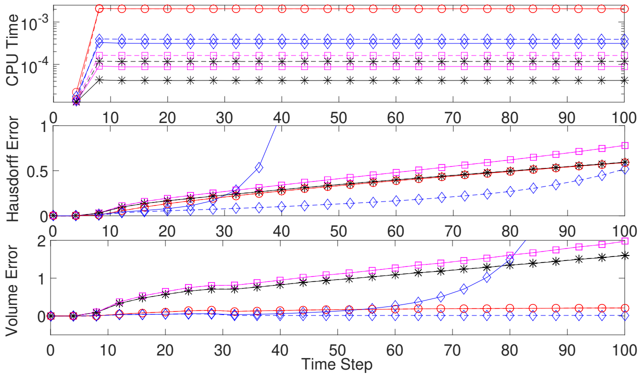

Figure 5.8: Average time and errors for reducing $\hat{X}_k$ in (5.10) to $o = 8$ with Methods 1–4 (□,⋆,○,◇, solid lines) and Methods 1'–4' (□,⋆,○,◇, dashed lines). Errors are relative to the true reachable set, $\Theta_V = [v(\hat{X}_k)^{1/n} - v(R_k)^{1/n}]/v(R_k)^{1/n}$ and $\Theta_H = d_H(\hat{X}_k, R_k)/r(R_k)$. After $k = 30$, $v(R_k)$ is too difficult to compute and is replaced by the minimum volume among the eight computed enclosures. All results are averaged over enough random systems to achieve a coefficient of variance $\mu/\sigma \leq 0.1$; i.e., 500 for CPU time, 1900 for Θ_V with $k \leq 30$, and 10^4 for Θ_H and Θ_V with $k > 30$.

5.5 Conclusions

Our results show that Method 4 is similar to Method 3 in terms of accuracy, but has much lower theoretical and empirical complexity, allowing it to address zonotopes up to at least 100 dimensions. Compared to the simpler Methods 1–2, Method 4 is more complex by a factor of n and empirically slower by 10–100×. However, it has lower volume and Hausdorff errors when reducing by a single order, but interestingly, only lower volume error when reducing by multiple orders in sufficiently high dimensions. In the context of reachability analysis, Method 4 is the most accurate over short horizons, but eventually becomes unstable. This instability is not well understood, but is largely mitigated by Method 4' and will be the subject of future investigations. Moreover, for applications in which the reachable set can be periodically intersected with measurements, long-term stability may be less important than short-term accuracy. When provided with equal CPU time, Methods 1–2 can accommodate much higher order zonotopes than Method 4', leading to lower Hausdorff errors. However, Method

4' still achieves lower volume errors. Moreover, many applications require operations that scale poorly with zonotope order, which may make Method 4' more desirable. For example, the approximate intersection of a zonotope with a bounded-error measurement in [37] scales as $n^5 n_g^2$, the active fault detection method in [66] scales exponentially in n_g, and computing volumes and half-space representations of zonotopes both scale as $\binom{n_g}{n}$ [64].

CHAPTER 6

GUARANTEED SAFE PATH AND TRAJECTORY TRACKING VIA REACHABILITY ANALYSIS USING DIFFERENTIAL INEQUALITIES

6.1 Introduction

This chapter presents rigorous nonlinear reachable set bounding methods for rapidly and accurately verifying the safety of automated vehicles tracking reference paths or trajectories under uncertainty. Path and trajectory tracking is important in automated driving systems for road vehicles, motion planning for autonomous robots, etc. [67, 68]. However, the reference paths and trajectories computed by such systems, which are safe by design, are not followed exactly by the vehicle due to various uncertainties in the vehicle's dynamics and environment (e.g., model parameters, tire slip, wind, measurement noises, etc.). These deviations of the real trajectory from the desired trajectory can lead to collisions or violations of other safety constraints. Therefore, methods for ensuring safety of a vehicle's real trajectory in real time are essential for achieving safe autonomous systems in practice. For example, such methods will be necessary to realize the anticipated safety benefits of autonomous road vehicles that result from eliminating delayed reactions and other human errors [67].

The existing literature on vehicle safety verification addresses several distinct problems based on how the vehicle's control inputs are handled. One class of methods assumes the inputs obey a probability distribution modeling the action of human drivers and aims to compute the likelihood of a collision [69, 70, 71]. These methods are primarily designed to generate warning alarms for human drivers, not for use in automated control systems. The computation of an optimal (i.e., safest) set of inputs for all vehicles in a road scene is discussed in [69], but safety is only ensured for nominal vehicle dynamics with no uncertainty.

A second class of methods that is more relevant for autonomous vehicle control treats the inputs as degrees of freedom and aims to compute either a feedback law or an open-loop input that guarantees safe trajectories [72, 73, 74, 75]. General approaches in this category require the solution of Hamilton-Jacobi-Isaac (HJI) partial differential equations, which is prohibitive in many cases because it scales exponentially in the number of states. This is partially addressed by dimension reduction methods in [73], but remains a significant limitation.

A third class of methods considers the simpler problem of verifying safety for a fixed control input specified *a priori*. This input can be specified as either an open-loop input [76, 77, 78, 79] or a fixed feedback law [80, 34, 81, 10, 82, 83, 84]. The most popular feedback approach is to first compute a safe reference trajectory (or path) and then follow it using a closed-loop tracking controller. Although these methods only assess the safety of a given control input rather than synthesizing a safe input, they address a critical subtask that can be used within larger algorithms for synthesizing safe controllers or motion plans. The methods in [76, 77, 79, 83, 84] compute the probability of safety violations by sampling or using stochastic reachable sets. Therefore, these methods cannot make rigorous safety guarantees, which is a drawback in some applications. Moreover, sampling-based methods are computationally demanding for systems with more than a few uncertain quantities, which limits their use for online safety verification. In contrast, the methods in [78, 80, 34, 81, 10, 82] aim to provide rigorous safety guarantees for systems subject to bounded uncertainties using reachability analysis techniques. However, efficiently computing an accurate enclosure of the reachable set of a nonlinear system is a significant challenge. To avoid this, most safety verification approaches use linear models [78, 81] or linearizations of nonlinear models [34, 74, 80]. Unfortunately, verifying safety of a linearized model does not ensure that the original model is safe. To date, the only guaranteed safety verification approach applicable to nonlinear vehicle models is given in [10, 82]. For the example considered in [10], it was shown that this method can verify the safety of a trajectory about 2× faster than the real vehicle traverses the trajectory. While this is promising, there is still a need for significantly more efficient methods to support verification for more complex models and to enable the use of online verification within iterative algorithms for safe controller synthesis. In practice, autonomous vehicles often update their trajectories every few milliseconds [85, 86, 87], so reachability-based verification on a similar time-scale is desirable.

This chapter focuses on the problem of rigorous safety verification for nonlinear vehicle models under a fixed feedback controller. Specifically, given a vehicle 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. We are interested in rigorous enclosures because they can be used to ensure safety of the planned path or trajectory with certainty by subsequently testing for intersections with obstacles or other unsafe sets. Moreover, we consider verification of a fixed tracking controller, rather than the more challenging problem of safe controller synthesis, because effective tracking controllers are available and widely used for many vehicle models, and we expect that a technology for efficiently verifying their safety in real-time will be very useful within practical iterative approaches for safe controller synthesis and motion planning.

Many methods are available for computing rigorous reachable set enclosures for continuous-time nonlinear systems. However, these methods often exhibit an unworkable compromise between accuracy and computational efficiency, particularly for systems with strong nonlinearities or large uncertainties. The zonotope-based method in [88], which has been applied for safety verification in [82, 10], propagates valid enclosures over discrete time steps using a conservative linearization technique with rigorously bounded linearization errors. Although this method is effective in many cases, the linearization error bound can be conservative for systems with strong nonlinearities. Moreover, high-order zonotopes and/or partitioning may be required to achieve high accuracy, which may become inefficient. Another class of reachability methods propagates valid enclosures over discrete time steps by first constructing a Taylor expansions of the states with respect to time and then computing rigorous bounds on the coefficients and remainder term [89]. Early methods computed these bounds using interval arithmetic, but contemporary methods achieve much higher accuracy using Taylor model arithmetic, which is based on multivariate Taylor expansions with respect to uncertain parameters [90, 91, 92, 93]. However, high accuracy may require high-order Taylor models, which also comes with high computational cost.

A final class of reachability methods is based on the theory of differential inequalities (DI). These methods compute valid 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 [94]. This is very efficient, which is attractive for online verification, but it usually computes very conservative bounds. Several more recent DI methods have addressed this by replacing intervals with polytopes [27], Taylor models [95], or mean value enclosures [96]. These methods produce much tighter bounds than standard DI, but are not as efficient. Another category of DI methods aims to use model redundancy to mitigate the conservatism of the standard interval 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 within iterative refinement algorithms to tighten the bounds continuously as they are propagated forward in time [24, 26, 117]. Importantly, this method can be applied to general nonlinear systems that do not satisfy any known invariants by *manufacturing* invariants [26]. This process involves embedding the system within a higher-dimensional system that obeys invariants by design (see §6.2 for details). 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 [24, 26, 117]. However, this approach requires significant problem insight to apply effectively, especially when invariants must be manufactured. To date, successful strategies have only been clearly demonstrated for models that arise from dynamic mass and energy balances, particularly in the chemical engineering domain, where it is relatively straightforward to manufacture simple and effective affine invariants.

In this chapter, we demonstrate the application of advanced redundancy-based DI methods to three representative case studies in vehicle path and trajectory tracking. The application of redundancy-based DI to this class of problems is challenging for three primary reasons. First, to the best of our knowledge, the models we consider do not naturally obey any invariants. Moreover, compared to mass and energy balance models, it much more difficult to identify effective manufactured invariants. Second, the presence of a feedback law in these models causes a significant interval dependency problem, which leads to very conservative bounds using interval-based methods if it is not addressed. Both of these challenges are explained in more detail in §6.2. Finally, the vehicle models of interest involve several functions that do not have well-defined interval evaluations, or whose interval evaluations violate Lipschitz regularity conditions that are required by DI-based reachability methods.

To address these issues, we first develop extended interval operations for several functions common in vehicle models and prove Lipschitz regularity. 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. Moreover, we develop highly effective nonlinear manufactured invariants. In all cases, we ultimately obtain reachability bounds that are greatly improved relative to the standard DI method, and appear both accurate and efficient enough to support many online safety verification tasks, although there is clearly still room for improvement. Finally, we conclude with a discussion of lessons learned and general strategies that are likely to be effective for other path and trajectory tracking problems.

6.1.1 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 $\kappa : D_\kappa \subset \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_u}$, and consider the following closed-loop system with input $\mathbf{u}$, disturbance $\mathbf{w}$, and state $\mathbf{x}$:

$$\dot{\mathbf{x}}(t) = \mathbf{f}_0(t, \mathbf{x}(t), \mathbf{w}(t), \mathbf{u}(t)), \text{ a.e. } t \in I, \tag{6.1a}$$

$$\mathbf{u}(t) = \kappa(t, \mathbf{x}(t), \mathbf{w}(t)), \tag{6.1b}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0. \tag{6.1c}$$

We are interested in computing reachability bounds for systems of the form (6.1) under a given path or trajectory tracking controller κ. We assume throughout that all states can be measured exactly, and we allow κ to depend on $\mathbf{w}(t)$ to account for cases where some disturbances are also measured. In practice, κ will also depend on a fixed reference path or trajectory, but we suppress this dependence for brevity. To further simplify notation, we define the closed-loop right-hand side $\mathbf{f} : D_f \subset \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$ by $\mathbf{f}(t, \mathbf{z}, \mathbf{v}) \equiv \mathbf{f}_0(t, \mathbf{z}, \mathbf{v}, \kappa(t, \mathbf{z}, \mathbf{v}))$. Then, (6.1) is equivalent to

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{w}(t)), \text{ a.e. } t \in I, \tag{6.2a}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0. \tag{6.2b}$$

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 time-varying uncertainties or disturbances as

$$\mathcal{W} \equiv \{\mathbf{w} \in (L^1(I))^{n_w} : \mathbf{w}(t) \in W \text{ for a.e. } t \in I\}. \tag{6.3}$$

Similarly, let $X_0 \subset \mathbb{R}^{n_x}$ be a compact interval of admissible initial conditions. Let $\mathcal{AC}(I, \mathbb{R}^n)$ denote the space of absolutely continuous functions from I into $\mathbb{R}^n$. We assume that (6.2) has a unique solution $\mathbf{x} \in \mathcal{AC}(I, \mathbb{R}^{n_x})$ corresponding to every $(\mathbf{x}_0, \mathbf{w}) \in X_0 \times \mathcal{W}$, and we denote this solution by $\mathbf{x}(\cdot; \mathbf{x}_0, \mathbf{w})$ when explicit dependence on $(\mathbf{x}_0, \mathbf{w})$ is necessary for clarity.

Definition 6. The *reachable set* of (6.2) is defined for every $t \in I$ by

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

Moreover, functions $\mathbf{x}^L, \mathbf{x}^U : I \to \mathbb{R}^{n_x}$ are called *state bounds* for (6.2) if $\text{Re}(t) \subset [\mathbf{x}^L(t), \mathbf{x}^U(t)]$, $\forall t \in I$.

]

The objective of this chapter is to compute state bounds for closed-loop path and trajectory problems of the form (6.2) that are both accurate and efficient enough to support rigorous motion planning and real-time safety verification tasks.

6.1.2 Notation

For $\mathbf{z}^L, \mathbf{z}^U \in \mathbb{R}^n$, let $Z = [\mathbf{z}^L, \mathbf{z}^U]$ denote the compact n-dimensional 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 $Z \subset D$. Let $\mathbb{IR}^+$ denote the set of all intervals Z such that $Z \subset \mathbb{R}^+$. Let $\mathbf{h} : D \subset \mathbb{R}^n \to \mathbb{R}^m$. 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 $X \in D_H$. A function $\mathbf{h}$ is called *factorable* if it can be written explicitly as a finite composition of elementary operations such as binary addition, binary multiplication, and intrinsic univariate functions ($-x$, x^n, e^x, etc.). For any *factorable* function $\mathbf{h}$, a specific inclusion function called the *natural interval extension* of $\mathbf{h}$ can be readily computed using interval arithmetic [101].

The space $\mathbb{IR}^n$ is a metric space under the Hausdorff metric $d_H(Z_1, Z_2) = \max\{\|\mathbf{z}_1^L - \mathbf{z}_2^L\|_\infty, \|\mathbf{z}_1^U - \mathbf{z}_2^U\|_\infty\}$ [101]. Then, following standard metric space definitions, the open ball of radius $\epsilon > 0$ centered at $X \in \mathbb{IR}^n$ is defined by $B_\epsilon(X) \equiv \{Z \in \mathbb{IR}^n : d_H(X, Z) < \epsilon\}$. A set $D \subset \mathbb{IR}$ is open if for every $Z \in D$ there exists a $\epsilon > 0$ such that $B_\epsilon(Z) \subset D$. Moreover, a function $F : D \subset \mathbb{IR}^n \to \mathbb{IR}^m$ is locally Lipschitz continuous on D if for every $Z \in D$, there exist constants $M, \epsilon > 0$ such that $d_H(F(X), F(Y)) < M d_H(X, Y)$ for every $X, Y \in B_\epsilon(Z) \cap D$.

6.2 Differential Inequalities

This section introduces the differential inequalities (DI) methods that will be used to compute reachability bounds for the closed loop system (6.2). We assume throughout that we have an inclusion function for $\mathbf{f}$ in (6.2). We denote this function by $F : D_F \subset \mathbb{I}D_f \to \mathbb{IR}$ and further denote $[\mathbf{f}^L(X), \mathbf{f}^U(X)] = F(X)$. Such a function can be computed, e.g., using interval arithmetic. We also require the following interval functions, which select an individual face of a given interval.

Definition 7. 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

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

The standard DI method originally proposed in [94] computes state bounds $X(t) \equiv$

$[\mathbf{x}^L(t), \mathbf{x}^U(t)]$ as the solutions of the following system of ODEs:

$$\begin{aligned}
\dot{x}_i^L(t) &= f_i^L([t,t], \beta_i^L\left(X(t)\right), W), \\
\dot{x}_i^U(t) &= f_i^U([t,t], \beta_i^U\left(X(t)\right), W), \\
X(t_0) &= X_0.
\end{aligned} \tag{6.4}$$

To understand this method, note that at the initial time we have $\mathbf{x}(t_0) \in X_0 = [\mathbf{x}^L(t_0), \mathbf{x}^U(t_0)]$. In order for, e.g., the i^{th} lower bound $x_i^L(t)$ 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 corresponding to any $(\mathbf{x}_0, \mathbf{w}) \in X_0 \times \mathcal{W}$; i.e. $\dot{x}_i^L(t) \leq \dot{x}_i(t)$. However, more careful analysis shows that it is really 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 (6.4).

As discussed in Section 6.1, the standard DI method is very efficient, but often gives very conservative bounds [24]. 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, consider the ODE $\dot{x}_1 = f_1(\mathbf{x}) = -ax_1x_2 + bx_2x_3$, in which x_2 appears twice. If the inclusion function F_1 is computed using interval arithmetic, then it will bound the range of f_1 assuming that these two instances of x_2 are independent, leading to overestimation. The dependency problem 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 such as mean value forms, although this is less efficient. In the example above, the problem can also be eliminated by simply rewriting f_1 as $f_1(\mathbf{x}) = (-ax_1 + bx_3)x_2$ and applying interval arithmetic to this factored expression. Notably, substantially different bounds can be obtained from DI using different expressions of $\mathbf{f}$, even though these expressions are equivalent in real number arithmetic. Rearrangements like this are important for getting good results from interval methods, but good rearrangements are not always possible.

A more subtle and often more significant source of conservatism in DI is the *historical dependency problem* [26]. 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 the range of $\mathbf{f}$ also leads to overestimation. Historical dependency is a weakness of DI itself and cannot be resolved by refactoring $\mathbf{f}$ or using more sophisticated inclusion functions. In fact, it would persist even if F returned the interval hull of the range of $\mathbf{f}$ over any interval of interest. Historical dependency can be mitigated by propagating non-interval reachable set enclosures such as polytopes or Taylor models because such enclosures can capture some of the dependence between state variables. However, such methods lose much of the speed that is so attractive in interval methods.

To address these limitations, several papers have subsequently developed efficient, interval-based DI methods that compute much tighter bounds than standard DI by exploiting redundant model equations [24, 26, 117]. By redundant model equations, we refer to any relationships between the states of a system that are known *a priori* to be satisfied by all solutions of the system. Examples of redundant equations that are often satisfied in applications are non-negativity of certain states, conservation mass, energy, or chemical species [24], the unit norm of rotation quaternions in some vehicle models [118], and various other solution invariants. Such relationships are useful because they provide information about the historical dependency between system states that is not captured by the standard DI method. In redundancy-based DI methods, these relationships are used to limit the range of inputs over which each f_i must be bounded when computing the right-hand sides of the bounding ODEs (6.4), often leading to much tighter bounds.

In what follows, we assume that redundant information is available in the form of an *a priori* enclosure G (see Assumption 5) and present the main details of the method in [117] for exploiting this enclosure to achieve tighter bounds. Subsequently, we will discuss how this approach can be applied to general systems for which no *a priori* enclosure is known using the concept of *manufactured invariants*.

Assumption 5. 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 (6.2) with $(\mathbf{x}_0, \mathbf{w}) \in X_0 \times \mathcal{W}$ satisfies $(t, \mathbf{x}(t), \mathbf{w}(t)) \in G$ for all $t \in I$.

The method in [117] makes use of G through a special kind of inclusion function for $\mathbf{f}$ called $\mathcal{R}$. The function $\mathcal{R}$ takes t and intervals Z and V as input and 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 the conventional inclusion function used in standard DI, which computes an interval enclosure of $\mathbf{f}(t, \mathbf{z}, \mathbf{v})$ for all $(\mathbf{z}, \mathbf{v}) \in Z \times V$. The inclusion function $\mathcal{R}$ also needs to satisfy several technical conditions detailed in the following formal assumption.

Assumption 6. Let $\mathcal{R} : D_{\mathcal{R}} \subset \mathbb{R} \times \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_w} \to \mathbb{IR}^{n_x}$ be an interval function satisfying:

1. For any $(t, Z, V) \in D_{\mathcal{R}}$, the set $\{t\} \times Z \times V$ is contained in the domain of $\mathbf{f}$, D_f, and

$$\begin{aligned}\mathcal{R}(t, Z, V) \supset \{\sigma \in \mathbb{R}^{n_x} : \sigma = \mathbf{f}(t, \mathbf{z}, \mathbf{v}),\ (\mathbf{z}, \mathbf{v}) \in Z \times V,\\ (t, \mathbf{z}, \mathbf{v}) \in G\}.\end{aligned} \tag{6.5}$$

2. $D_{\mathcal{R}}$ is open with respect to t and Z. Specifically, for every $(\hat{t}, \hat{Z}, \hat{V}) \in D_{\mathcal{R}}$, there exists $\epsilon > 0$ such that $(t, Z, \hat{V}) \in D_{\mathcal{R}}$ for every $t \in B_\epsilon(\hat{t})$ and $Z \in B_\epsilon(\hat{Z})$.

3. $\mathcal{R}$ is locally Lipschitz continuous with respect to Z, uniformly with respect to t.

Specifically, for any $(\hat{t}, \hat{Z}, \hat{V}) \in \mathcal{R}$, there exists ϵ, $L > 0$ such that

$$d_H(\mathcal{R}(t, Z, \hat{V}), \mathcal{R}(t, \bar{Z}, \hat{V})) \leq L d_H(Z, \bar{Z}), \tag{6.6}$$

for every $t \in B_\epsilon(\hat{t})$ and $Z, \bar{Z} \in B_\epsilon(\hat{Z})$.

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

Given any $\mathcal{R}$ satisfying Assumption 6, state bounds for (6.2) can be computed using the following corollary from [117].

Corollary 5. Suppose that $\mathbf{x}^L, \mathbf{x}^U \in \mathcal{AC}(I, \mathbb{R}^{n_x})$ are solutions of the following system of ODEs with $i \in \{1, \ldots, n_x\}$ and $X(t) \equiv [\mathbf{x}^L(t), \mathbf{x}^U(t)]$:

$$\begin{aligned} \dot{x}_i^L(t) &= \mathcal{R}_i^L(t, \beta_i^L(X(t)), W), && (6.7) \\ \dot{x}_i^U(t) &= \mathcal{R}_i^U(t, \beta_i^U(X(t)), W), && \\ X(t_0) &= X_0. && (6.8) \end{aligned}$$

Then, for every $(\mathbf{x}_0, \mathbf{w}) \in X_0 \times \mathcal{W}$, the solution $\mathbf{x}(\cdot; \mathbf{x}_0, \mathbf{w}) \in \mathcal{AC}(I, \mathbb{R}^{n_x})$ of (6.2) satisfies $\mathbf{x}(t; \mathbf{x}_0, \mathbf{w}) \in X(t)$ for all $t \in I$.

Remark 6. The version of Corollary 5 given above is a simplified version of the more general result proven in [117]. Specifically, in [117], the assumption that $(t, \mathbf{x}(t), \mathbf{w}(t)) \in G$ is generalized to $(t, \mathbf{x}(t), \mathbf{w}(t), \dot{\mathbf{x}}(t)) \in G$, with G now a subset of $\mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \times \mathbb{R}^{n_x}$, and the inclusion property of $\mathcal{R}$ is generalized to $\mathcal{R}(t, Z, V) \supset \{\sigma \in \mathbb{R}^{n_x} : \sigma = \mathbf{f}(t, \mathbf{z}, \mathbf{v}), (\mathbf{z}, \mathbf{v}) \in Z \times V, (t, \mathbf{z}, \mathbf{v}, \sigma) \in G\}$. Since none of the redundant relationships defining G in our case studies depend on $\dot{\mathbf{x}}(t)$, we have omitted the $\dot{\mathbf{x}}(t)$ dependence above for simplicity.

To implement Corollary 5 numerically, a specific inclusion function $\mathcal{R}$ must be defined. In [117], a general approach is proposed consisting of two steps. Given generic inputs $(t, Z, V) \in D_{\mathcal{R}}$, the method first refines the intervals Z and V by eliminating regions that violate the constraint $(t,\mathbf{z},\mathbf{v}) \in G$. Specifically, this step results in refined 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, these refined intervals are used to evaluate a standard inclusion function for $\mathbf{f}$; i.e., $\mathcal{R}(t,Z,V) = F([t,t],Z^\dagger,V^\dagger)$. In the first step, the refined intervals $Z^\dagger$ and $V^\dagger$ are computed using a variant of the interval Krawczyk method [101] called the κ-operator. This method is applicable whenever G can be written in the general form

$$G = \{(t,\mathbf{z},\mathbf{v}) \in D_G : \mathbf{g}(t,\mathbf{z},\mathbf{v}) \leq \mathbf{0}, \mathbf{h}(t,\mathbf{z},\mathbf{v}) = \mathbf{0}\}, \tag{6.9}$$

where $(\mathbf{g},\mathbf{h}) : D_G \subset \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_g} \times \mathbb{R}^{n_h}$ are locally Lipschitz continuous functions.

Although this redundancy-based DI method provides much tighter state bounds than standard DI in many cases, the key drawback is that it only applies to systems for which redundant information is available in the form of an *a priori* enclosure G. To address this, Shen and Scott [26] developed the concept of *manufactured invariants*, which extends the redundancy-based DI approach to general systems for which no *a priori* enclosure is known. To present this idea in a sufficiently general form, assume that the uncertainty $\mathbf{w}(t)$ in (6.2) can be decomposed into two parts, $\mathbf{w}(t) = (\mathbf{d}(t),\mathbf{p})$, 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 $\phi : D_\phi \subset \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_z}$ and defining the new state variables $\mathbf{z}(t;\mathbf{x}_0,\mathbf{w}) \equiv \phi(\mathbf{x}(t;\mathbf{x}_0,\mathbf{w}),\mathbf{p})$. The choice of ϕ is discussed further below.

Next, the new states are differentiated to form the augmented system

$$\begin{aligned}
\dot{\mathbf{x}}(t) &= \mathbf{f}(t, \mathbf{x}(t), \mathbf{w}(t)), \\
\dot{\mathbf{z}}(t) &= \frac{\partial \phi}{\partial \mathbf{x}}(\mathbf{x}(t), \mathbf{p})\mathbf{f}(t, \mathbf{x}(t), \mathbf{w}(t)), \\
\mathbf{x}(t_0) &= \mathbf{x}_0, \\
\mathbf{z}(t_0) &= \phi(\mathbf{x}_0, \mathbf{p}).
\end{aligned} \tag{6.10}$$

We assume that (6.10) has a unique solution $(\mathbf{x}, \mathbf{z}) \in \mathcal{AC}(I, \mathbb{R}^{n_x+n_z})$ corresponding to every $(\mathbf{x}_0, \mathbf{w}) \in X_0 \times \mathcal{W}$. For any $(\mathbf{x}_0, \mathbf{w}) \in X_0 \times \mathcal{W}$, if $(\mathbf{x}, \mathbf{z}) \in \mathcal{AC}(I, \mathbb{R}^{n_x+n_z})$ is the solution of (6.10), then $\mathbf{x} \in \mathcal{AC}(I, \mathbb{R}^{n_x})$ must be the solution of (6.2). Therefore, to bound the reachable set of (6.2), it suffices to compute state bounds for (6.10). But, by design, all solutions of (6.10) are guaranteed to satisfy the *manufactured invariants* $\mathbf{z}(t) - \phi(\mathbf{x}(t), \mathbf{p}) = \mathbf{0}$, $\forall t \in I$. Thus, state bounds can be computed by applying Corollary 5 with $G \equiv \{(t, (\mathbf{x}, \mathbf{z}), (\mathbf{d}, \mathbf{p})) : \mathbf{z} - \phi(\mathbf{x}, \mathbf{p}) = \mathbf{0}\}$.

This technique has been shown to result in much tighter bounds than standard DI for many problems with no known *a priori* enclosure. However, achieving good results requires careful choice of the function ϕ defining the new states. The aim is to choose ϕ such that the function $\frac{\partial \phi}{\partial \mathbf{x}}\mathbf{f}$ appearing in the ODEs for $\mathbf{z}$ reduces to an expression that is simple in the sense that it does not suffer much from the dependency problems discussed above. For example, for a two-dimensional system described by $\dot{x}_1 = -x_1 - r(\mathbf{w}, \mathbf{x})$ 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 (6.7) in Corollary 5 are solved for the augmented system, simplifications of this sort can cause the bounds on $\mathbf{z}$ to accumulate conservatism less quickly than those on $\mathbf{x}$, 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 $\mathcal{R}$, which can slow or prevent conservatism from accumulating in the bounds of $\mathbf{x}$ as well. A more detailed explanation of the effects of using manufactured invariants can be found in [26]. Although the example above is contrived, similar term cancellations and other simplifications can be achieved by choosing simple affine ϕ functions in a wide variety of practical examples, and the improvements in bound accuracy relative to standard DI are stark [26]. However, most of these examples are drawn from (bio)chemical engineering applications and the models, which are derived from differential balances on mass, energy, and chemical species, share some advantageous structural features. Thus, while this technique is broadly applicable in principle, effective strategies for choosing ϕ have so far only been demonstrated for a limited class of models.

In the remainder of this chapter, we aim to apply the advanced redundancy-based DI method formalized in Corollary 5 to obtain accurate reachability bounds for some representative path and trajectory tracking problems. In preparation for this, we conclude this section by highlighting some key challenges posed by this class of problems in light of the discussion above. The first is that the systems of interest are closed-loop, with right-hand sides of the form

$$\mathbf{f}(t,\mathbf{z},\mathbf{v}) = \mathbf{f}_0(t,\mathbf{z},\mathbf{v},\kappa(t,\mathbf{z},\mathbf{v})). \tag{6.11}$$

Regardless of the functional forms of $\mathbf{f}_0$ and κ, this structure ensures that there is a significant interval dependency problem due to the two appearances of $\mathbf{z}$ and $\mathbf{v}$. Thus, if interval arithmetic is applied directly to $\mathbf{f}$ in this form to evaluate the inclusion function $\mathcal{R}$, the result will almost certainly be very conservative. In particular, 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, to the best of our knowledge, the systems we consider do not satisfy any known *a priori* enclosures. It is therefore necessary to manufacture invariants. However, unlike the chemical engineering examples mentioned above, there are no obvious choices of ϕ that lead to desirable simplifications when forming the augmented right-hand sides $\frac{\partial \phi}{\partial \mathbf{x}}\mathbf{f}$, so new strategies must be developed. Finally, the systems we consider involve several functions that do not have well-defined interval evaluations, or whose interval evaluations would violate the Lipschitz property of $\mathcal{R}$ required by Assumption 6. Therefore, new interval evaluations with the appropriate properties must be defined.

6.3 Interval Inclusion Functions for Some Non-Standard Functions

This section introduces interval inclusion functions for several functions that do not appear in standard interval arithmetic libraries. This includes the multi-valued inverses of x^2 and $\cos(x)$, which are required in our interval refinement algorithms, as well as the derivatives of some trigonometric functions, which appear commonly in the vehicle models of interest here. Each inclusion function is designed to ensure locally Lipschitz continuity, which will be needed in order to use them in the construction of an inclusion function $\mathcal{R}$ satisfying Assumption 6.

We begin with an inclusion function for the multi-valued square root defined on the positive reals, including both the positive and negative roots as illustrated in Figure 6.1. The standard interval inclusion function for this operation, which gives the exact interval hull of its range over any $X \in \mathbb{IR}$, is $\pm\sqrt{X} = [-\sqrt{x^U}, +\sqrt{x^U}]$. However, this definition inherits the non-locally-Lipschitz behavior of $\sqrt{x}$ at $x = 0$. To avoid this, we define a weaker modified inclusion function using upper and lower linearizations around $x = \epsilon > 0$, as shown in Figure 6.1. For intervals with $x^U \geq \epsilon$, the modified inclusion function returns $[-\sqrt{x^U}, +\sqrt{x^U}]$ as usual. However, for intervals with $x^U < \epsilon$, the upper and lower bounds are instead determined by the upper and lower linearizations.

Definition 8. For any tolerance $\epsilon > 0$, define $\bar{\sqrt{\ }} : \mathbb{IR}^+ \to \mathbb{IR}$ by

$$\bar{\sqrt{\ }}X = \begin{cases} \left[-\sqrt{x^U}, \sqrt{x^U}\right] & \text{if } x^U \geq \epsilon, \\ \left[-\frac{1}{2\sqrt{\epsilon}}x^U - \frac{\sqrt{\epsilon}}{2}, \frac{1}{2\sqrt{\epsilon}}x^U + \frac{\sqrt{\epsilon}}{2}\right] & \text{if } x^U < \epsilon. \end{cases} \tag{6.12}$$

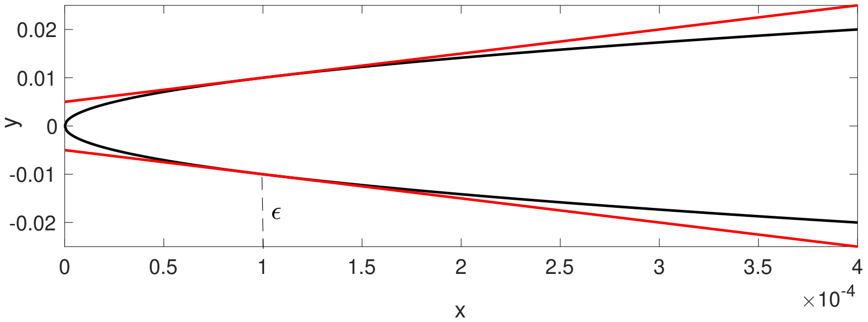

Figure 6.1: Multi-valued square root $y = \pm\sqrt{x}$ (black) with lower and upper linearizations at $\epsilon = 10^{-4}$, $y = -\frac{1}{2\sqrt{\epsilon}}x - \frac{\sqrt{\epsilon}}{2}$ and $y = \frac{1}{2\sqrt{\epsilon}}x + \frac{\sqrt{\epsilon}}{2}$ (red).

The next two theorems show that $\bar{\sqrt{\ }}$ is a valid inclusion function for $\pm\sqrt{x}$ and is Lipschitz continuous on $\mathbb{IR}^+$.

Theorem 9. *If* $X \in \mathbb{IR}^+$ *and* $x \in X$*, then* $\pm\sqrt{x} \in \bar{\sqrt{\ }}X$*.*

Proof. Choose any $X \in \mathbb{IR}^+$ and any $x \in X$. Monotonicity of the square root function implies that $\sqrt{x} \in \left[\sqrt{x^L}, \sqrt{x^U}\right]$ and $-\sqrt{x} \in \left[-\sqrt{x^U}, -\sqrt{x^L}\right]$ [102]. Thus, $\pm\sqrt{x} \in \left[\sqrt{x^L}, \sqrt{x^U}\right] \cup \left[-\sqrt{x^U}, -\sqrt{x^L}\right] \subset \left[-\sqrt{x^U}, \sqrt{x^U}\right]$. Therefore, if $x^U \geq \epsilon$, then $\pm\sqrt{x} \in \bar{\sqrt{\ }}X$. Suppose instead that $x^U < \epsilon$. Since the square root function is concave, $\sqrt{x^U}$ must be dominated by the linearization $\sqrt{\epsilon} + \frac{1}{2\sqrt{\epsilon}}(x^U - \epsilon) = \frac{1}{2\sqrt{\epsilon}}x^U + \frac{\sqrt{\epsilon}}{2}$. Therefore, $\pm\sqrt{x} \in \left[-\sqrt{x^U}, \sqrt{x^U}\right]$ implies $\pm\sqrt{x} \in \left[-\frac{1}{2\sqrt{\epsilon}}x^U - \frac{\sqrt{\epsilon}}{2}, \frac{1}{2\sqrt{\epsilon}}x^U + \frac{\sqrt{\epsilon}}{2}\right] = \bar{\sqrt{\ }}X$. $\square$

Theorem 10. *The extended square root function is Lipschitz continuous on* $\mathbb{IR}^+$*.*

Proof. Choose any $X_1, X_2 \in \mathbb{IR}^+$. First, assume that $x_1^U, x_2^U \geq \epsilon$. By the definition of the Hausdorff distance,

$$d_H(\sqrt[-]{X_1},\ \sqrt[-]{X_2}) = \left|\sqrt{x_1^U} - \sqrt{x_2^U}\right|.$$

Let $L \geq \max_{y \geq \epsilon} \frac{1}{2\sqrt{y}} = \frac{1}{2\sqrt{\epsilon}}$. Then, the Mean Value Theorem gives $\left|\sqrt{x_1^U} - \sqrt{x_2^U}\right| \leq L\left|x_1^U - x_2^U\right|$. Therefore, $d_H(\sqrt[-]{X_1},\ \sqrt[-]{X_2}) \leq L d_H\left(X_1, X_2\right)$.

Next, assume that $x_1^U, x_2^U \leq \epsilon$. By the definition of Hausdorff distance,

$$d_H\left(\sqrt[-]{X_1},\ \sqrt[-]{X_2}\right) = \frac{1}{2\sqrt{\epsilon}}\left|x_1^U - x_2^U\right| \leq L d_H\left(X_1, X_2\right).$$

Finally, assume w.l.o.g. that $x_1^U \geq \epsilon$ and $x_2^U \leq \epsilon$. Then,

$$d_H\left(\sqrt[-]{X_1},\ \sqrt[-]{X_2}\right) = \left|\sqrt{x_1^U} - \left(\frac{1}{2\sqrt{\epsilon}}x_2^U + \frac{\sqrt{\epsilon}}{2}\right)\right|. \tag{6.13}$$

The term inside the absolute value above must be nonnegative since $\sqrt{x_1^U}$ dominates $\sqrt{\epsilon}$, $\sqrt{\epsilon}$ is equal to the linearization evaluated at ϵ, $\frac{1}{2\sqrt{\epsilon}}\epsilon + \frac{\sqrt{\epsilon}}{2}$, and this in turn dominates $\frac{1}{2\sqrt{\epsilon}}x_2^U + \frac{\sqrt{\epsilon}}{2}$ by monotonicity of the linearization. Therefore,

$$\begin{aligned}
& d_H\left(\sqrt[-]{X_1},\ \sqrt[-]{X_2}\right) = \sqrt{x_1^U} - \frac{1}{2\sqrt{\epsilon}}x_2^U - \frac{\sqrt{\epsilon}}{2}, \qquad (6.14)\\
&= \sqrt{x_1^U} - \frac{1}{2\sqrt{\epsilon}}x_2^U - \frac{\sqrt{\epsilon}}{2} - \frac{1}{\sqrt{x_1^U}}(x_1^U - x_2^U) + \frac{1}{\sqrt{x_1^U}}(x_1^U - x_2^U),\\
&= \left(\frac{1}{\sqrt{x_1^U}} - \frac{1}{2\sqrt{\epsilon}}\right)x_2^U - \frac{\sqrt{\epsilon}}{2} + \frac{1}{\sqrt{x_1^U}}(x_1^U - x_2^U).
\end{aligned}$$

Since $x_1^U \geq \epsilon$, we have $\frac{1}{\sqrt{x_1^U}} \leq \frac{1}{\sqrt{\epsilon}}$. Thus,

$$d_H\left(\sqrt[-]{X_1},\ \sqrt[-]{X_2}\right) \leq \frac{1}{2\sqrt{\epsilon}}x_2^U - \frac{\sqrt{\epsilon}}{2} + \frac{1}{\sqrt{\epsilon}}(x_1^U - x_2^U). \tag{6.15}$$

Since $x_2^U \leq \epsilon$, the term $\frac{1}{2\sqrt{\epsilon}}x_2^U - \frac{\sqrt{\epsilon}}{2} \leq 0$. Therefore,

$$d_H\left(\bar{\sqrt{X_1}}, \bar{\sqrt{X_2}}\right) \leq \frac{1}{\sqrt{\epsilon}}(x_1^U - x_2^U) \leq 2L d_H(X_1, X_2).$$

Therefore, the Lipschitz constant $2L$ is valid on all of $\mathbb{IR}^+$. $\square$

Next, we develop an inclusion function of arcsin. The function arcsin is monotonically increasing, but is not Lipschitz continuous at $x = -1$ and $x = 1$, as shown in Figure 6.2. In this case, it suffices for our purposes in §6.4 to avoid this non-Lipschitz behavior by simply restricting the domain of our inclusion function to intervals contained in the open interval $(-1, 1)$ and establish local Lipschitz continuity on this domain.

Definition 9. Let $D \equiv \{x \in \mathbb{R} : -1 < x < 1\}$ and define the $\bar{\arcsin} : \mathbb{I}D \to \mathbb{IR}$ by

$$\bar{\arcsin}(X) = [\arcsin(x^L), \arcsin(x^U)]. \tag{6.16}$$

Theorem 11. *For any $X \in \mathbb{I}D$ and $x \in X$, $\arcsin(x) \in \bar{\arcsin}(X)$.*

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

Theorem 12. *The function $\bar{\arcsin}$ is locally Lipschitz continuous on $\mathbb{I}D$.*

Proof. Consider the two real-valued functions $lb(x^L, x^U) = \arcsin(x^L)$ and $ub(x^L, x^U) = \arcsin(x^U)$ describing the upper and lower bounds in (6.16). According to Theorem 2.5.30 in [104], $\bar{\arcsin}$ is locally Lipschitz continuous on $\mathbb{I}D$ if an only if both lb and ub are locally Lipschitz continuous on $\{(x^L, x^U) \in D \times D : x^L \leq x^U\}$. But this follows directly from the fact that arcsin is continuously differentiable, and hence locally Lipschitz continuous, on $D = (-1, 1)$. $\square$

Next, we develop an inclusion function for the multi-valued arccos function including both positive and negative branches as illustrated in Figure 6.2 (middle). It is easy to see from the figure that $\arccos(X) = [-\arccos(x^L), \arccos(x^L)]$ is a valid inclusion function. However, this inclusion function inherits non-locally-Lipschitz behavior from the real-valued arccos function at -1 and 1. For our purposes in §6.4, it suffices to simply exclude -1 from the domain of our inclusion function, as was done with arcsin above. However, we will need to apply this inclusion function to intervals that potentially contain 1. Therefore, we propose a weaker inclusion function that makes use of upper and lower linearizations at a point ϵ arbitrarily close to 1 (see Figure 6.2).

Definition 10. Let $D \equiv (-1, 1]$, choose any $\epsilon \in D$, and define $\mathrm{arc\bar{c}os} : \mathbb{I}D \to \mathbb{IR}$ by

$$\mathrm{arc\bar{c}os}(X) = \begin{cases} \Big[-\arccos(x^L), \arccos(x^L)\Big] & \text{if } x^L \leq \epsilon, \\ \Big[\frac{1}{\sqrt{1-\epsilon^2}}(x^L - \epsilon) - \arccos(\epsilon), & \\ -\frac{1}{\sqrt{1-\epsilon^2}}(x^L - \epsilon) + \arccos(\epsilon)\Big] & \text{if } x^L > \epsilon. \end{cases} \tag{6.17}$$

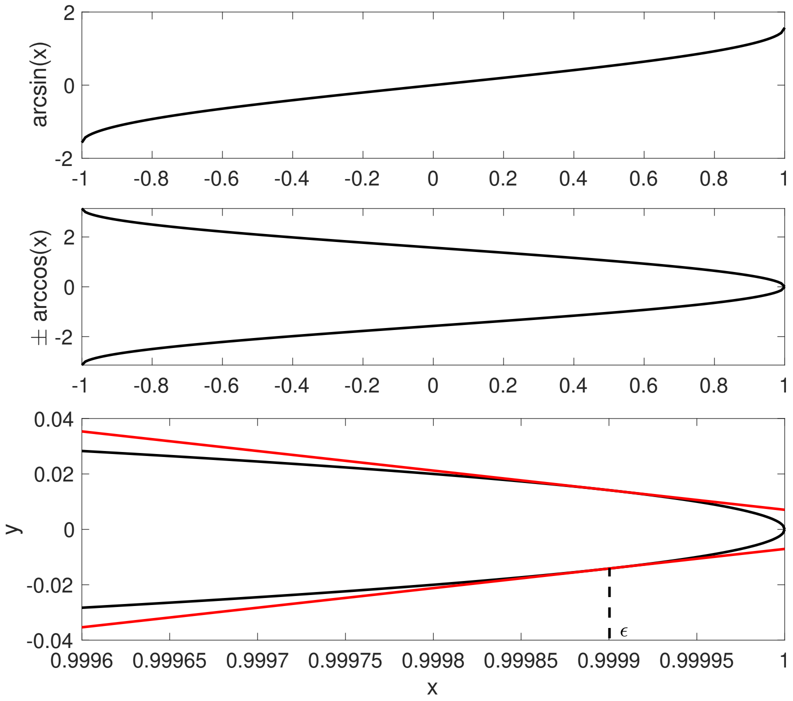

Figure 6.2: The arcsin function (top), the multi-valued arccos function on $[-1, 1]$ (middle), and the multi-valued arccos function near $x = 1$ with lower and upper linearizations at $\epsilon = 0.9999$, $y = \pm\left(\frac{1}{\sqrt{1-\epsilon^2}}(y^L - \epsilon) - \arccos(\epsilon)\right)$(red).

Theorem 13. *For any $X \in \mathbb{I}D$ and $x \in X$, $\pm\arccos(x) \in \bar{\arccos}(X)$.*

Proof. Choose any $\in \mathbb{I}D$ and any $x \in X$. Since arccos is monotonically decreasing on $(-1, 1]$, we have $\arccos(x^U) \leq \arccos(x) \leq \arccos(x^L)$ and $-\arccos(x^L) \leq -\arccos(x) \leq -\arccos(x^U)$. Combining these inequalities with $-\arccos(x^L) \leq 0 \leq \arccos(x^U)$, we conclude that

$$\pm\arccos(x) \subset \left[-\arccos(x^L), \arccos(x^L)\right]. \tag{6.18}$$

This proves that $\pm\arccos(x) \in \bar{\arccos}(X)$ provided that $x^L \leq \epsilon$.

Assume instead that $\epsilon < x^L \leq 1$. To show that $\pm\arccos(x) \in \bar{\arccos}(X)$ in this case, it suffices to prove that

$$\arccos(x^L) \leq -\frac{1}{\sqrt{1-\epsilon^2}}(x^L - \epsilon) + \arccos(\epsilon). \tag{6.19}$$

It is equivalent to prove that $-\frac{1}{\sqrt{1-\epsilon^2}}x^L + \frac{\epsilon}{\sqrt{1-\epsilon^2}} + \arccos(\epsilon) - \arccos(x^L) \geq 0$. We construct a function $f(x) = \frac{x}{\sqrt{1-\epsilon^2}} + \arccos(x)$, which is monotonically decreasing for $x \in [\epsilon, 1]$. Thus, we have $f(\epsilon) \geq f(x^L)$ since $\epsilon < x^L \leq 1$. Therefore, Therefore, we have $\arccos(x^L) \leq -\frac{1}{\sqrt{1-\epsilon^2}}x^L + \frac{\epsilon}{\sqrt{1-\epsilon^2}} + \arccos(\epsilon)$. ☐

Theorem 14. *The function* $\bar{\arccos}$ *is locally Lipschitz continuous on* $\mathbb{I}D$*.*

Proof. Let $ub : D \times D \to \mathbb{R}$ be the real-valued function corresponding to the upper bound in (6.17); i.e.,

$$ub(x^L, x^U) = \begin{cases} \arccos(x^L) & \text{if } x^L \leq \epsilon, \\ \frac{-1}{\sqrt{1-\epsilon^2}}(x^L - \epsilon) + \arccos(\epsilon) & \text{if } x^L > \epsilon. \end{cases} \tag{6.20}$$

Let lb denote the lower bounding function defined analogously. According to Theorem 2.5.30 in [104], $\bar{\arccos}$ is locally Lipschitz continuous on $\mathbb{I}D$ if both lb and ub are locally Lipschitz continuous on $D \times D$. We prove this below for ub. The proof for lb is analogous.

Since ub is independent of x^U for this inclusion function, we may view it as a univariate function and show that it is locally Lipschitz on D. Choose any $\hat{x} \in D$. We must show that there exists $\eta, L > 0$ such that

$$|ub(x_1) - ub(x_2)| \leq L|x_1 - x_2|, \quad \forall x_1, x_2 \in B_\eta(\hat{x}) \cap D. \tag{6.21}$$

Choose any $\eta > 0$ small enough that the closure of $B_\eta(\hat{x})$ does not contain -1 and define

$$L_1 = \sup\left\{\left|\frac{d}{dx}\arccos(x)\right| : x \in B_\eta(\hat{x}) \cap (-1, \epsilon]\right\}, \tag{6.22}$$

$$= \sup\left\{\frac{1}{\sqrt{1-x^2}} : x \in B_\eta(\hat{x}) \cap (-1, \epsilon]\right\}. \tag{6.23}$$

This constant is finite because neither 1 nor -1 is a limit point of $B_\eta(\hat{x}) \cap (-1, \epsilon]$. Next, define $L_2 = \frac{1}{\sqrt{1-\epsilon^2}}$ and let $L = \max(L_1, L_2)$. We will show that (6.21) holds with this L.

Choose any $x_1, x_2 \in B_\eta(\hat{x}) \cap D$. If $x_1, x_2 \leq \epsilon$, then by the Mean Value Theorem,

$$|ub(x_1) - ub(x_2)| = |\arccos(x_1) - \arccos(x_2)|, \tag{6.24}$$

$$\leq L_1|x_1 - x_2|. \tag{6.25}$$

Similarly, if $x_1, x_2 > \epsilon$, then

$$|ub(x_1) - ub(x_2)| = \left|\frac{-1}{\sqrt{1-\epsilon^2}}(x_1 - \epsilon) - \frac{-1}{\sqrt{1-\epsilon^2}}(x_2 - \epsilon)\right|,$$

$$\leq L_2|x_1 - x_2|. \tag{6.26}$$

Finally, assume w.l.o.g. that $x_1 \leq \epsilon$ and $x_2 > \epsilon$. Then,

$$\begin{aligned}
&|ub(x_1) - ub(x_2)| \qquad (6.27)\\
&= \left|\arccos(x_1) - \left(\frac{-1}{\sqrt{1-\epsilon^2}}(x_2 - \epsilon) + \arccos\epsilon\right)\right|,\\
&\leq |\arccos(x_1) - \arccos(\epsilon)| + \left|\frac{1}{\sqrt{1-\epsilon^2}}(x_2 - \epsilon)\right|,\\
&\leq L_1\,|x_1 - \epsilon| + L_2\,|x_2 - \epsilon|,\\
&\leq L\,|x_1 - x_2|.
\end{aligned}$$

Thus, (6.21) holds. □

Finally, we define locally Lipschitz inclusion functions for the following four trigonometric functions, which commonly appear in vehicle models.

Definition 11. 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

$$h_1(x) = \begin{cases} \frac{\cos x - 1}{x} & x \neq 0, \\ 0 & x = 0. \end{cases} \tag{6.28}$$

$$h_2(x) = \begin{cases} \frac{\sin x}{x} & x \neq 0, \\ 1 & x = 0. \end{cases} \tag{6.29}$$

$$h_3(x) = \begin{cases} \frac{\cos x - 1}{x^2} & x \neq 0, \\ -0.5 & x = 0. \end{cases} \tag{6.30}$$

$$h_4(x) = \begin{cases} \frac{x\cos(x) - \sin(x)}{x^2} & x \neq 0, \\ 0 & x = 0. \end{cases} \tag{6.31}$$

The functions h_1–h_4 are shown in Figure 6.3. In the following definitions, mid denotes the function that returns the middle value of its three arguments; i.e., $\mathrm{mid}(-10, 5, 4) = 4$.

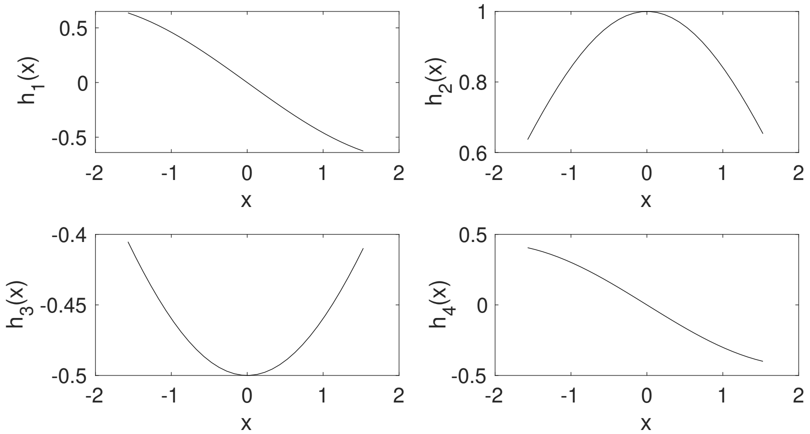

Figure 6.3: The functions h_1–h_4 defined in Definition 11

Definition 12. 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

$$H_1(X) = [h_1(x^U), h_1(x^L)], \tag{6.32}$$

$$H_2(X) = [\min(h_2(x^L), h_2(x^U)), h_2(\text{mid}(x^L, 0, x^U))], \tag{6.33}$$

$$H_3(X) = [h_3(\text{mid}(x^L, 0, x^U)), \max(h_3(x^L), h_3(x^U))], \tag{6.34}$$

$$H_4(X) = [h_4(x^U), h_4(x^L)]. \tag{6.35}$$

Theorem 15. *For any $X \in \mathbb{I}D$ and $x \in X$, we have $h_1(x) \in H_1(X)$, $h_2(x) \in H_2(X)$, $h_3(x) \in H_3(X)$, $h_4(x) \in H_4(X)$.*

Proof. Choose any $X \in \mathbb{I}D$ and $x \in X$. The inclusions $h_1(x) \in H_1(X)$ and $h_4(x) \in H_4(X)$ follow from the fact that h_1 and h_4 are monotonically decreasing functions. The inclusion function for h_2 is based on the fact that h_2 has maximum at $x = 0$, is monotonically increasing on $(-\pi/2, 0)$, and is monotonically decreasing on $(0, \pi/2)$. Therefore, the lower bound is the smaller value among $h_2(x^L)$ and $h_2(x^U)$. The maximum value will be $h_2(x^U)$ if $x^U < 0$, $h_2(x^L)$ if $x^L > 0$, and $h_2(0)$ if $x^L < 0 < x^U$. Therefore, the upper bound is always attained at $\text{mid}(x^L, 0, x^U)$. The proof for H_3 is analogous. $\square$

Theorem 16. *The functions H_1, H_2, H_3, and H_4 are locally Lipschitz continuous on $\mathbb{I}D$.*

Proof. It is straightforward to show that h_1–h_4 are continuously differentiable functions (in fact, h_3 and h_4 are the derivatives of h_1 and h_2). Therefore, h_1–h_4 are locally Lipschitz continuous on $(-\pi/2, \pi/2)$. Consider the real-valued functions $lb_1(x^L, x^U)$–$lb_4(x^L, x^U)$ and $ub_1(x^L, x^U)$–$ub_4(x^L, x^U)$ defined as the lower and upper bound functions of H_1–H_4. According to Theorem 2.5.30 in [104], H_1–H_4 are locally Lipschitz continuous on $\mathbb{I}D$ if and only if $lb_1(x^L, x^U)$–$lb_4(x^L, x^U)$ and $ub_1(x^L, x^U)$–$ub_4(x^L, x^U)$ are locally Lipschitz continuous on $\{(x^L, x^U) \in D \times D : x^L \leq x^U\}$. But this follows immediately from local Lipschitz continuity of h_1–h_4, min, max, and mid (Lemma 2.5.25 in [104]). $\square$

We close this section by recalling the extended intersection defined in [104], which will be needed in the refinement algorithms developed in the next section.

Definition 13. Define the *extended intersection* $\bar{\cap} : \mathbb{IR}^n \times \mathbb{IR}^n \to \mathbb{IR}^n$ componentwise by

$$(X \bar{\cap} Z)_i = \left[\text{mid}\left(x_i^L, x_i^U, z_i^L\right), \text{mid}\left(x_i^L, x_i^U, z_i^U\right)\right], \tag{6.36}$$

for all $i \in \{1, \ldots, n\}$.

Note that $(X \bar{\cap} Z) = (X \cap Z)$ whenever $(X \cap Z)$ is nonempty. The following regularity result is from Lemma 2.8 in [46].

Theorem 17. *The extended intersection $\bar{\cap}$ is Lipschitz continuous on $\mathbb{IR}^n$ with Lipschitz constant 1.*

6.4 Case Studies

In this section, we apply the advanced redundancy-based DI method formalized in Corollary 5 to obtain accurate reachability bounds for two trajectory tracking problems and one path tracking problem. The first and third examples consider a simple Dubins car model using two 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 from the perspective of reachability and rigorous safety verification [67]. The second example considers trajectory tracking for a more complex full size vehicle model, where both turning rate and acceleration are limited. All case studies were implemented in C++ on a laptop with a 2.9 GHz Intel Core i7, and ODEs were solved using CVODE with default settings [119].

Example 1. Consider the following vehicle dynamics, where (x, y) is the vehicle position, v is the velocity, θ is the heading angle, and ω is the heading rate:

$$\begin{aligned} \dot{x} &= v\cos(\theta), \qquad (6.37) \\ \dot{y} &= v\text{sin}(\theta), \\ \dot{\theta} &= \omega. \end{aligned}$$

Define the reference trajectory $x_{ref}, y_{ref}, \theta_{ref} : I \to \mathbb{R}$ with $I = [0, 8]$ s as the solution of (6.37) with the piecewise constant control inputs v_{ref} and ω_{ref} given in Table 6.1. The control objective is to manipulate v and ω to bring the vehicle's real trajectory close to this reference. To do this, we use the control law from [120], which

Table 6.1: Reference control inputs for Example 1

Time interval (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 interval (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

is defined in terms of the following error coordinates:

$$\begin{aligned} x_e &= \cos(\theta)(x_{ref} - x) + \sin(\theta)(y_{ref} - y), \\ y_e &= -\sin(\theta)(x_{ref} - x) + \cos(\theta)(y_{ref} - y), \\ \theta_e &= \theta_{ref} - \theta. \end{aligned} \tag{6.38}$$

The control law is given by

$$\begin{aligned} \omega &= \omega_{ref} + v_{ref}(k_2 y_e + k_3 \sin(\theta_e)) + d_1, \\ v &= v_{ref}\cos(\theta_e) + k_1 x_e + d_2, \end{aligned} \tag{6.39}$$

with gains $k_1 = 10$ 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 [120] but are assumed to corrupt the desired control inputs here. We assume that these disturbances are time-invariant and satisfy $d_1 \in [-0.1, 0.1]$ rad/s and $d_2 \in [-1, 1]$ cm/s. Moreover, we assume that the initial condition is uncertain and satisfies $\mathbf{x}_0 \in X_0 = [-5, 5] \times [-5, 5] \times [-\pi/6, \pi/6]$. In [120], it is shown that the vehicle trajectory under this control law converges to the reference trajectory when $d_1 = d_2 = 0$. However, with nonzero disturbances d_1 and d_2, the vehicle may not converge to the reference trajectory exactly. Therefore, we aim to compute a rigorous enclosure of the real state trajectories.

The most straightforward approach to compute interval bounds on the states x,

y, and θ is to apply the standard DI method directly to (6.37) with the control law (6.39). As discussed in §6.2, this requires an inclusion function for the closed-loop right-hand side functions, 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 (6.38) in interval arithmetic. Then, bounds on the control inputs V and Ω are computed using (6.39). Finally, the right-hand sides of (6.37) are evaluated in interval arithmetic. The result of applying standard DI with this inclusion function are shown in Figure 6.4 (green) along with 500 sampled trajectories generated by solving the closed-loop system with $\mathbf{x}_0$, d_1, and d_2 drawn from uniform distributions over their interval bounds (gray).

Clearly, the bounds are very conservative. This is largely due to a significant dependency problem in the inclusion function described above, as discussed in Section 6.2. Specifically, θ affects the right-hand sides of x and y in two ways - directly through (6.37) and again through the control input v using (6.39) and (6.38). In real arithmetic, this allows the controller to cancel out the systems natural dynamics and impose the desired behavior. However, when we apply interval arithmetic to bound the closed-loop right-hand side function, the instance of θ in the original dynamics is treated as independent from that in the control law. Hence, the state bounds on (6.37) explode quickly.

To mitigate this dependency problem, a better approach is to apply the standard DI method to the dynamics of the error coordinates (6.38) rather than to the original coordinates in (6.37):

$$\begin{aligned}
\dot{x}_e &= \omega y_e - v + v_{ref}\cos\theta_e, \qquad (6.40)\\
\dot{y}_e &= -\omega x_e + v_{ref}\sin(\theta_e),\\
\dot{\theta}_e &= \omega_{ref} - \omega.
\end{aligned}$$

Plugging in the control law (6.39) and simplifying gives the closed-loop error dynamics

$$\begin{aligned}
\dot{x}_e &= \left(\omega_{ref} + v_{ref}\left(k_2 y_e + k_3 \text{sin}\left(\theta_e\right)\right) + d_1\right) y_e - k_1 x_e - d_2, \\
\dot{y}_e &= -\left(\omega_{ref} + v_{ref}(k_2 y_e + k_3 \text{sin}(\theta_e)) + d_1\right) x_e + v_{ref} \text{sin}(\theta_e), \\
\dot{\theta}_e &= -v_{ref}(k_2 y_e + k_3 \text{sin}(\theta_e)) - d_1. \qquad (6.41)
\end{aligned}$$

Applying DI to this system 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. As a specific example, note that the nonlinear term $v_{ref} \cos \theta_e$ is completely cancelled from the right-hand side function for x_e when the control law is substituted in. Since this simplification can be done analytically, before the use of interval arithmetic, the dependency problem discussed above is reduced. Once bounds on the error coordinates are computed, they can be mapped back to the original coordinates by evaluating the following inverse coordinate transformation in interval arithmetic:

$$\begin{aligned}
x &= x_{ref} - \cos(\theta_{ref} - \theta_e) x_e + \sin(\theta_{ref} - \theta_e) y_e, \qquad (6.42) \\
y &= y_{ref} - \sin(\theta_{ref} - \theta_e) x_e - \cos(\theta_{ref} - \theta_e) y_e, \\
\theta &= \theta_{ref} - \theta_e.
\end{aligned}$$

Figure 6.4 shows the results of this approach (blue). As expected, the results are significantly tighter those obtained by applying DI directly to (6.37). However, the bounds are still very conservative and diverge quickly.

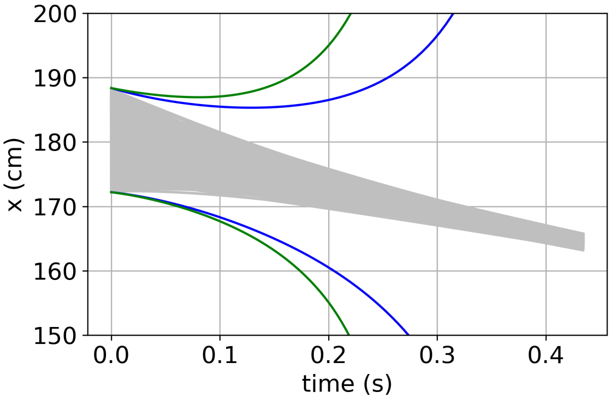

Figure 6.4: Example 1: Bounds on the vehicle position produced by applying standard DI to (6.37) (green) and (6.41) (blue) with 500 sampled trajectories (gray).

To achieve further improvements, we now manufacture invariants for (6.41) following the method in [26]. As described in §6.2, the aim is to find a C^1 function ϕ of the system states 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 (6.41) shows that this cannot be done using any affine ϕ, as is the case for most models considered in [26]. Specifically, although there are common nonlinear terms among the ODEs in (6.41) that would be advantageous to eliminate in $\frac{\partial \phi}{\partial \mathbf{x}}\mathbf{f}$ (e.g. $(\omega_{ref} + v_{ref}\,(k_2 y_e + k_3 \sin{(\theta_e)})))$, they cannot be cancelled out by any linear combination of these ODEs. Therefore, we need to manufacture invariants using nonlinear combinations of the states. For reasons discussed below, an excellent candidate is the following Lyapunov function for (6.41), which was used to prove stability in [120]:

$$\mathcal{V} = \frac{1}{2}(x_e^2 + y_e^2) + \frac{(1 - \cos(\theta_e))}{k_2}. \tag{6.43}$$

To use this function as a manufactured invariant, we define $\mathcal{V}$ as a new state variable and augment (6.41) with the ODE derived by differentiating $\mathcal{V}$ with respect to time. The ODE obtained in this way benefits from several term cancellations and algebraic simplifications, ultimately leading to the form

$$\dot{\mathcal{V}} = -k_1 x_e^2 - \frac{v_{ref} k_3 \sin^2(\theta_e)}{k_2} - x_e d_2 + \frac{\sin(\theta_e) d_1}{k_2}. \tag{6.44}$$

Let $\mathbf{z} \equiv (x_e, y_e, \theta_e, \mathcal{V})$ and $\mathbf{p} = (d_1, d_2)$ be shorthand for generic augmented state and uncertain parameter vectors. Then, by definition, the augmented system consisting of (6.41) and (6.44) satisfies Assumption 5 with the *a priori* enclosure

$$G = \left\{ (t, \mathbf{z}, \mathbf{p}) \in \mathbb{R}^7 : \mathcal{V} = \frac{1}{2}(x_e^2 + y_e^2) + \frac{(1 - \cos(\theta_e))}{k_2} \right\}. \tag{6.45}$$

To apply the redundancy-based DI method in Corollary 5 using this G, it remains to define an inclusion function $\mathcal{R}$ satisfying Assumption 6. Recall from §6.2 that the general approach in [117] defines $\mathcal{R}$ in two steps. Given generic inputs $(t, Z, P) \in D_{\mathcal{R}}$, the method first computes a refined interval $Z^{\dagger} \times P^{\dagger}$ such that $\{t\} \times Z^{\dagger} \times P^{\dagger}$ contains $(\{t\} \times Z \times P) \cap G$, and then bounds $\mathbf{f}$ over $\{t\} \times Z^{\dagger} \times P^{\dagger}$. In [117], $Z^{\dagger} \times P^{\dagger}$ is computed using a variant of the interval Krawczyk method [101].

Although this general approach can be used here, we instead define a more effective custom refinement algorithm based on direct algebraic rearrangements of the manufactured invariant (6.43). Given any $(t, Z, P) \in \mathbb{R} \times \mathbb{IR}^4 \times \mathbb{IR}^2$ and any $\mathbf{z} = (x_e, y_e, \theta_e, \mathcal{V}) \in Z$ and $\mathbf{p} = (d_1, d_2) \in P$ such that $(t, \mathbf{z}, \mathbf{p}) \in G$, the following rearrangements of (6.43) must hold:

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

Therefore, denoting Z component-wise by $Z = X_e \times Y_e \times \Theta_e \times V$, $\mathbf{z}$ must satisfy the following inclusions, where the right-hand-side are evaluated in interval arithmetic using the inclusion functions $\sqrt[-]{\ }$ and $\mathrm{ar\bar{c}cos}$ defined in §6.3:

$$\begin{aligned}
\mathcal{V} &\in \frac{1}{2}\left(X_e^2 + Y_e^2\right) + \frac{(1-\cos(\Theta_e))}{k_2}, \\
x_e &\in \sqrt[-]{2\left(V - \frac{(1-\cos(\Theta_e))}{k_2} - \frac{1}{2}Y_e^2\right)}, \\
y_e &\in \sqrt[-]{2\left(V - \frac{(1-\cos(\Theta_e))}{k_2} - \frac{1}{2}X_e^2\right)}, \\
\theta_e &\in \mathrm{ar\bar{c}cos}\left(1 - k_2\left(V - \frac{1}{2}(X_e^2 + Y_e^2)\right)\right).
\end{aligned} \tag{6.46}$$

The right-hand sides of these inclusions can be used to refine the intervals V, X_e, Y_e, and Θ_e, respectively, and this refinement can be done iteratively. Our proposed definition of $\mathcal{R}$ based on these refinements is given in Algorithm 8. The refinements are done in the loop beginning on line 3, while the final enclosure of $\mathbf{f}$ (i.e., the right-hand sides of (6.41) and (6.44)) is computed in lines 12–15. All set operations in Algorithm 8 are done using standard interval arithmetic or the operations defined in §6.3, and we choose the number of iterations as $l = 2$. A formal proof that this algorithm satisfies Assumption 6 is given at the end of this subsection.

Algorithm 8 An implementation of $\mathcal{R}$ for Example 1

```
function $\mathcal{R}(t, Z, P)$
    $(X_e, Y_e, \Theta_e, V) \leftarrow Z$, $(D_1, D_2) \leftarrow P$
    for $i = 1$ to $l$ do
        $V \leftarrow V \bar{\cap} \frac{1}{2}(X_e^2 + Y_e^2) + \frac{(1-\cos(\Theta_e))}{k_2}$
        $SQ_{X_e} \leftarrow X_e^2 \bar{\cap} \left(2(V - \frac{(1-\cos(\Theta_e))}{k_2} - \frac{1}{2}Y_e^2)\right)$
        $X_e \leftarrow X_e \bar{\cap} \sqrt[\bar{}]{SQ_{X_e}}$
        $SQ_{Y_e} \leftarrow Y_e^2 \bar{\cap} 2(V - \frac{(1-\cos(\Theta_e))}{k_2} - \frac{1}{2}X_e^2)$
        $Y_e \leftarrow Y_e \bar{\cap} \sqrt[\bar{}]{SQ_{Y_e}}$
        $COS_{\Theta_e} \leftarrow \cos(\Theta_e) \bar{\cap} \left(1 - k_2(V - \frac{1}{2}Y_e^2 - \frac{1}{2}X_e^2)\right)$
        $\Theta_e \leftarrow \Theta_e \bar{\cap} \mathrm{arc}\bar{\mathrm{c}}\mathrm{os}(COS_{\Theta_e})$
    end for
    $\Sigma_1 \leftarrow (\omega_{ref} + v_{ref}(k_2 Y_e + k_3 \sin\Theta_e) + D_1)Y_e - k_1 X_e - D_2$
    $\Sigma_2 \leftarrow -(\omega_{ref} + v_{ref}(k_2 Y_e + k_3 \sin\Theta_e) + D_1)X_e + v_{ref}\sin\Theta_e$
    $\Sigma_3 \leftarrow -v_{ref}(k_2 Y_e + k_3 \sin(\Theta_e)) - D_1$
    $\Sigma_4 \leftarrow -k_1 X_e^2 - \frac{v_{ref} k_3 \sin^2(\Theta_e)}{k_2} - X_e D_2 + \sin(\Theta_e) D_1 / k_2$
return $\Sigma \leftarrow (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4)$
end function
```

Figure 6.5 shows the bounds on the error states (x_e, y_e, θ_e) computed by applying standard DI to (6.41) (Method (i), blue) and by applying redundancy-based DI (Corollary 5) to the augmented system (6.41) and (6.44) with $\mathcal{R}$ defined by Algorithm 8 (Method (ii), green). Note that the non-smoothness of the bounding trajectories in some figures is caused by the piecewise constant inputs used to generate the reference trajectory. While the bounds computed by standard DI rapidly diverge to $\pm\infty$, the bounds computed using redundancy-based DI are much more accurate and diverge slowly if at all. This indicates that using the Lyapunov function as a manufactured invariant is very effective at mitigating the dependency problems discussed in §6.2. Figures 6.6–6.7 show the corresponding bounds on the original states (x, y, θ) obtained by evaluating the inverse coordinate transformation (6.42) in interval arithmetic. It can be seen that the accuracy of the redundancy-based DI method is retained in the original coordinates. Moreover, although these bounds certainly leave room for improvement, they do appear to be accurate enough to support some motion planning or collision avoidance tasks.

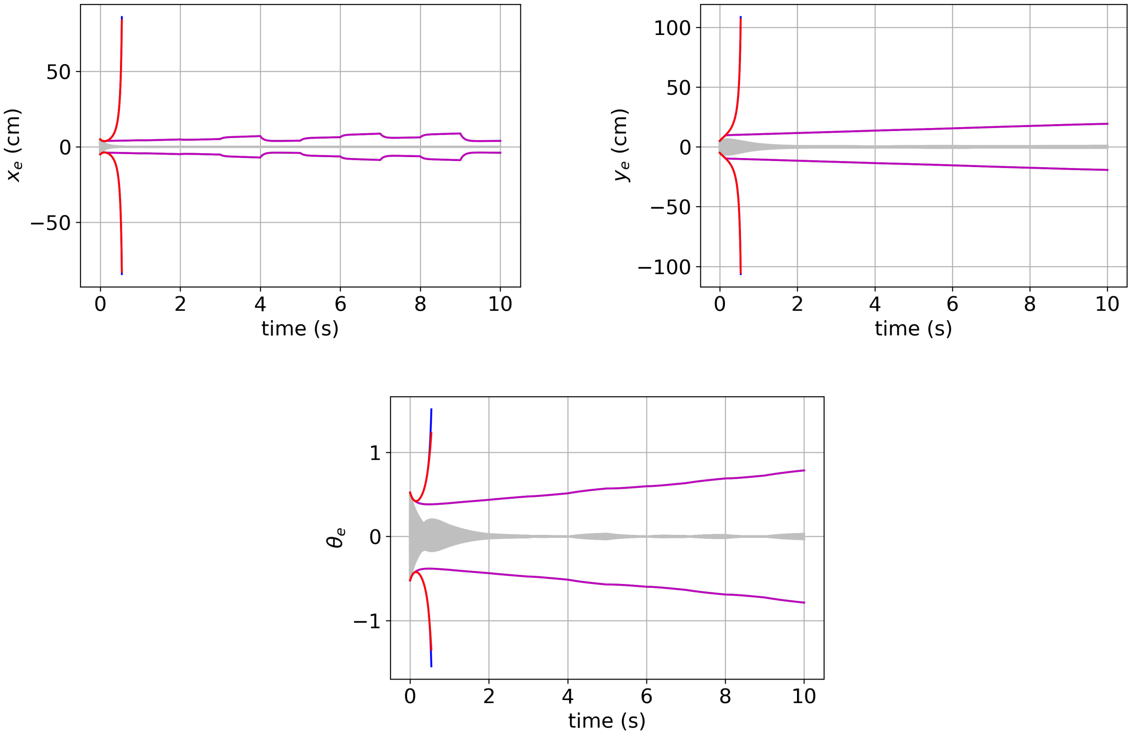

Figure 6.5: Example 1: Bounds on the error coordinates (x_e, y_e, θ_e) produced by (i) applying standard DI to (6.41) (blue), (ii) applying redundancy-based DI to (6.41) and (6.44) with manufactured invariant (6.43) (green), (iii) applying redundancy-based DI to (6.41), (6.44), and (6.37) with manufactured invariants (6.38), (6.42), and (6.43) (purple), and (iv) applying redundancy-based DI to (6.41) and (6.37) with manufactured invariants (6.38) and (6.42) (red) with 500 sampled trajectories (gray).

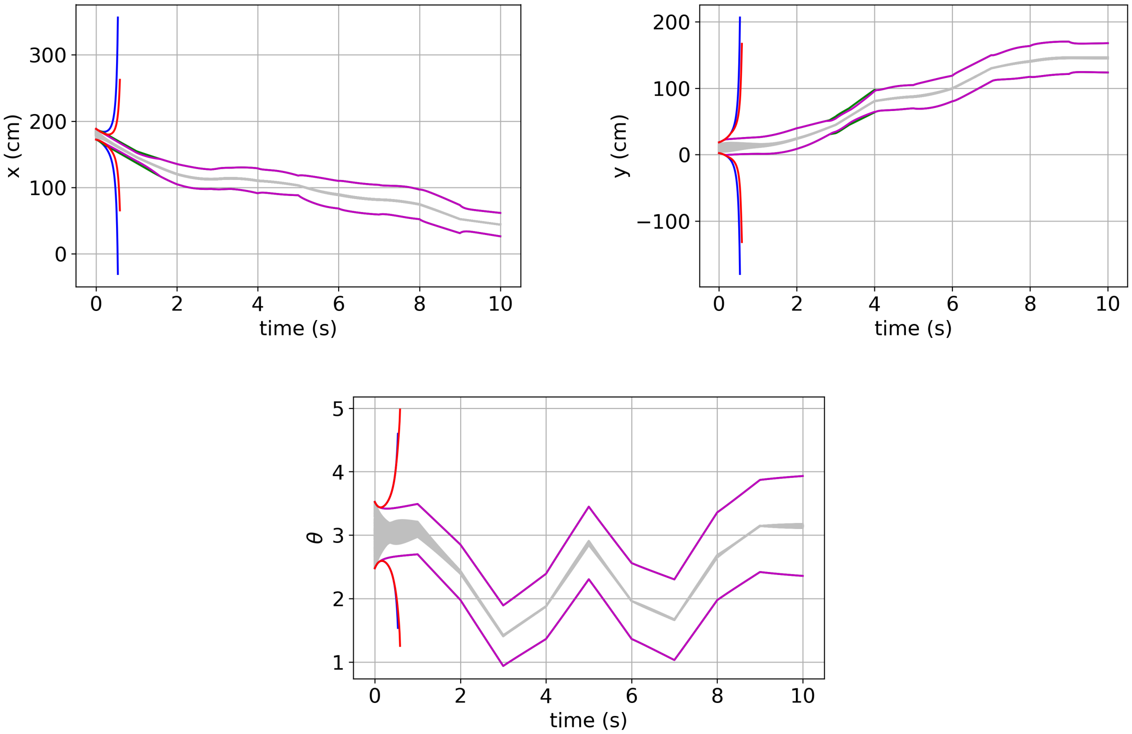

Figure 6.6: Example 1: Bounds on the original coordinates (x, y, θ) produced by (i) applying standard DI to (6.41) (blue), (ii) applying redundancy-based DI to (6.41) and (6.44) with manufactured invariant (6.43) (green), (iii) applying redundancy-based DI to (6.41), (6.44), and (6.37) with manufactured invariants (6.38), (6.42), and (6.43) (purple), and (iv) applying redundancy-based DI to (6.41) and (6.37) with manufactured invariants (6.38) and (6.42) (red) with 500 sampled trajectories (gray).

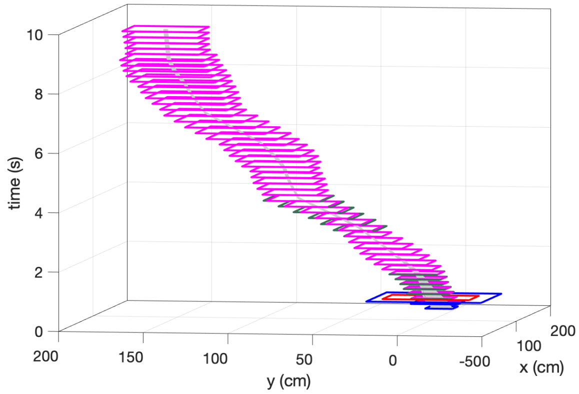

Figure 6.7: Example 1: Bounds on the vehicle positions produced by (i) applying standard DI to (6.41) (blue), (ii) applying redundancy-based DI to (6.41) and (6.44) with manufactured invariant (6.43) (green), (iii) applying redundancy-based DI to (6.41), (6.44), and (6.37) with manufactured invariants (6.38), (6.42), and (6.43) (purple), and (iv) applying redundancy-based DI to (6.41) and (6.37) with manufactured invariants (6.38) and (6.42) (red) with 500 sampled trajectories (gray).

In addition to using a Lyapunov function as a manufactured invariant, another potentially useful approach for introducing model redundancy into vehicle models is to write the model in multiple coordinate systems simultaneously. This could have advantages over using a single coordinate system if there are some aspects of the model are more simply represented in the first coordinate system and others that are more simply represented in the second. To try this approach for the present example, we now augment (6.41) and (6.44) with the closed-loop dynamics in the original coordinates described by (6.37) with (6.39). In addition to (6.43), the states of this augmented system also satisfy the invariants (6.38) and (6.42). Therefore, we can define the following *a priori* enclosure with $\mathbf{z} = (x_e, y_e, \theta_e, \mathcal{V}, x, y, \theta)$ and $\mathbf{p} = (d_1, d_2)$:

$$G = \Big\{(t, \mathbf{z}, \mathbf{p}) \in \mathbb{R}^{10} : (6.38),\ (6.42),\ \text{and}\ (6.43)\ \text{hold}\Big\}. \tag{6.47}$$

To apply the redundancy-based DI method in Corollary 5 using this G, we must again define an inclusion function $\mathcal{R}$ satisfying Assumption 6. We follow the same procedure as in Algorithm 8, but now include additional refinements based on (6.38) and (6.42). Given any $(t, Z, P) \in \mathbb{R} \times \mathbb{IR}^7 \times \mathbb{IR}^2$ and denoting Z component-wise by $Z = X_e \times Y_e \times \Theta_e \times V \times X \times Y \times \Theta$, (6.38) and (6.42) imply that any $\mathbf{z} \in Z$ and $\mathbf{p} \in P$ satisfying $(t, \mathbf{z}, \mathbf{p}) \in G$ must also satisfy the following inclusions:

$$\begin{aligned} x_e &\in \cos(\Theta)(X_{ref} - X) + \sin(\Theta)(Y_{ref} - Y), \\ y_e &\in -\sin(\Theta)(X_{ref} - X) + \cos(\Theta)(Y_{ref} - Y), \\ \theta_e &\in \theta_{ref} - \Theta, \\ x &\in X_{ref} - \cos(\theta_{ref} - \Theta_e)x_e + \sin(\theta_{ref} - \Theta_e)Y_e, \\ y &\in Y_{ref} - \sin(\theta_{ref} - \Theta_e)x_e - \cos(\theta_{ref} - \Theta_e)Y_e, \\ \theta &\in \theta_{ref} - \Theta_e. \end{aligned} \tag{6.48}$$

Our proposed definition of $\mathcal{R}$ based on these refinements is given in Algorithm 9. Compared to Algorithm 8, Algorithm 9 adds the refinements (6.48) in lines 5–10 and computes bounds on the right-hand sides of the ODEs (6.37) (which are now included in the augmented system) in lines 13–15. A formal proof that this algorithm satisfies Assumption 6 is given at the end of this subsection.

Algorithm 9 An implementation of $\mathcal{R}$ for Example 1 with additional invariants

```
**function** $\mathcal{R}(t, Z, P)$
    $(X_e, Y_e, \Theta_e, V, X, Y, \Theta) \leftarrow Z$ , $(D_1, D_2) \leftarrow P$
    **for** $i = 1$ to $l$ **do**
        Apply lines 4–10 in Algorithm 8
        $X \leftarrow X \bar{\cap} (x_{ref} - \cos(\Theta)X_e + \sin(\Theta)Y_e)$
        $Y \leftarrow Y \bar{\cap} (y_{ref} - \sin(\Theta)X_e - \cos(\Theta)Y_e)$
        $\Theta \leftarrow \Theta \bar{\cap} (\theta_{ref} - \Theta_e)$
        $X_e \leftarrow X_e \bar{\cap} (\cos(\Theta)(x_{ref} - X) + \sin(\Theta)(y_{ref} - Y))$
        $Y_e \leftarrow Y_e \bar{\cap} (-\sin(\Theta)(x_{ref} - X) + \cos(\Theta)(y_{ref} - Y))$
        $\Theta_e \leftarrow \Theta_e \bar{\cap} (\theta_{ref} - \Theta)$
    **end for**
    Apply lines 12–15 in Algorithm 8 to compute $\Sigma_1$–$\Sigma_4$
    $\Sigma_5 \leftarrow \cos(\Theta)(v_{ref}\cos(\Theta_e) + k_1 X_e + D_2)$
    $\Sigma_6 \leftarrow \sin(\Theta)(v_{ref}\cos(\Theta_e) + k_1 X_e + D_2)$
    $\Sigma_7 \leftarrow \omega_{ref} + v_{ref}(k_2 Y_e + k_3 \sin(\Theta_e)) + D_1$
**return** $\Sigma \leftarrow (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \Sigma_7)$
**end function**
```

The results applying redundancy-based DI (Corollary 5) with $\mathcal{R}$ defined by Algorithm 9 (Method (iii), purple) are shown in Figures 6.5–6.7. In the error coordinates, the bounds from this method lie entirely behind the plotted bounds for Method (ii), indicating that use of multiple coordinate systems to generate additional manufactured invariants offers no improvement over using just the Lyapunov function. In the original coordinates, Method (iii) offers a very slight improvement that can be seen, e.g., for y just before 4 s. We also compared the bounds obtained by using the manufactured invariants (6.38) and (6.42) (i.e., the coordinate transformations) without using the Lyapunov function (Method (iv), red). This resulted in diverging bounds that are only only slightly tighter than those of Method (i) (standard DI).

We conclude that the model redundancy offered by using both the original and error coordinates simultaneously is ineffective at mitigating the dependency problem for this example. We expect that using multiple coordinate systems will be effective in cases where each coordinate system is able to represent some aspect of the model more simply than the other. However, for this example, it appears that the error coordinates are universally better for interval computations, so that refinements based on the coordinate transformations (6.38) and (6.42) have the effect of using (X_e, Y_e, Θ_e) to tighten (X, Y, Θ), but rarely the reverse.

In terms of computational cost, standard DI is the most efficient method tested with a cost of 0.0009 s. However, this time is misleading because integration was stopped early due to divergence of the bounds. Method (ii) using the Lyapunov function requires 0.46 s, and Method (iii) using the Lyapunov function and the coordinate transformations requires 0.773 s. For context, it takes about 3 s to simulate 3125 real trajectories, which corresponds to a grid with only 5 values for each uncertain variable. Method (ii) clearly offers the best trade-off between accuracy and efficiency, producing effective bounds with a computational time that is equivalent to sampling about 480 real trajectories, and that is roughly 20× faster than the real travel time for this vehicle.

We close this subsection by proving that Algorithm 8 satisfies Assumption 6. The proof for Algorithm 9 is a straightforward extension of the same arguments and is omitted for brevity. Consider the augmented system consisting of (6.41) and (6.44) and let $\mathbf{z} = (x_e, y_e, \theta_e, \mathcal{V})$ and $\mathbf{p} = (d_1, d_2)$ denote generic state and parameter vectors. Similarly, let $Z = (X_e, Y_e, \Theta_e, V)$ and $P = (D_1, D_2)$ denote generic state and parameter interval vectors. Note that in verifying Assumption 6, we are free to choose the domain $D_{\mathcal{R}}$, provided that Algorithm 9 is well defined for any $(t, P, Z) \in D_{\mathcal{R}}$. However, it is desirable to choose $\mathcal{D}_{\mathcal{R}}$ as the largest set for which Assumption 6 holds because this maximizes the applicability of Corollary 5. Specifically, if the solutions of the bounding ODEs (6.7) leave $D_{\mathcal{R}}$, then they cease to exist as solutions of (6.7) and their validity is no longer ensured by Corollary 5. For this example, the only restriction we impose in the definition of $D_{\mathcal{R}}$ is that the interval Θ_e must be a subset of $(-\pi, \pi)$, which will be used to ensure that a domain violation does not occur in the $\bar{\arccos}$ function in line 10 of Algorithm 8. Since the θ_e bounds for Method (ii) in Figure 6.5 are well within $(-\pi, \pi)$ for all time, this requirement is not restrictive here.

Theorem 18. *Define* $D_{\mathcal{R}} \equiv \{(t, Z, P) \in \mathbb{R} \times \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_p} : \Theta_e \subset (-\pi, \pi)\}$. *Algorithm 8 is well defined for every* $(t, Z, P) \in D_{\mathcal{R}}$. *Moreover, Assumption 6 holds with* $\mathcal{R}(t, Z, P)$ *defined by Algorithm 8.*

Proof. Choose any $(t, Z, P) \in D_{\mathcal{R}}$. Algorithm 8 is well defined for (t, Z, P) if and only if no domain violations occur when evaluating the inclusion functions in lines 4–10 and lines 12–15. The only interval operations in these lines that could possibly lead to a domain violation (i.e., are not defined for every possible argument) are the $\bar{\sqrt{\ }}$ in lines 6 and 8 and the $\bar{\arccos}$ in line 10. But by the extended intersections in lines 5 and 7, SQ_{X_e} and SQ_{Y_e} are guaranteed to lie in X_e^2 and Y_e^2, respectively, and hence in $\mathbb{IR}^+$, which is the domain of $\bar{\sqrt{\ }}$. Regarding the $\bar{\arccos}$, first note that $(t, Z, P) \in D_{\mathcal{R}}$ implies that $\Theta_e \subset (-\pi, \pi)$ when line 9 is reached for the first time. Therefore, the extended intersection in 9 ensures that $COS_{\Theta_e} \subset \cos(\Theta_e) \subset (-1, 1]$. It follows that $\bar{\arccos}(COS_{\Theta_e})$ is well defined when line 10 is reached for the first time. Furthermore, the extended intersection in line 10 implies that Θ_e remains a subset of $(-\pi, \pi)$, so the same arguments apply in subsequent visits to lines 9–10. Therefore, $\mathcal{R}$ is well defined on $D_{\mathcal{R}}$.

To verify Condition 1 of Assumption 6, choose any $(t, Z, P) \in D_{\mathcal{R}}$ and let $\Sigma = \mathcal{R}(t, P, Z)$ be the output of Algorithm 8. We must show that

$$\Sigma \supset \{\mathbf{f}(t, \mathbf{z}, \mathbf{p}) : (\mathbf{z}, \mathbf{p}) \in Z \times P,\ (t, \mathbf{z}, \mathbf{p}) \in G\}, \tag{6.49}$$

where $\mathbf{f}$ denotes the right-hand sides of (6.41) and (6.44) and

$$G = \left\{ (t, \mathbf{z}, \mathbf{p}) \in \mathbb{R}^7 : \mathcal{V} = \frac{1}{2}(x_e^2 + y_e^2) + \frac{(1 - \cos(\theta_e))}{k_2} \right\}. \tag{6.50}$$

Choose any $(\mathbf{z}, \mathbf{p}) \in Z \times P$ satisfying $(t, \mathbf{z}, \mathbf{p}) \in G$ and let $(x_e, y_e, \theta_e, \mathcal{V}) = \mathbf{z}$. Since $\mathbf{z} \in Z$, the following inclusions all hold immediately after line 2: $x_e \in X_e$, $y_e \in Y_e$, $\theta_e \in \Theta_e$, and $\mathcal{V} \in V$. If these inclusions remain valid when line 12 is reached, then (6.49) must hold because lines 12–15 are direct interval evaluations of the right-hand sides of (6.41) and (6.44) (i.e., $\mathbf{f}$). Thus, it suffices to show that these inclusions are maintained through lines 4–10. Since $(t, \mathbf{z}, \mathbf{p}) \in G$, (6.46) must hold. But (6.46) implies that $\mathcal{V} \in V$ still holds after line 4. Similarly, (6.46) implies that $x_e^2 \in SQ_{X_e}$ after line 5. Thus, by Theorem 9, $x_e \in X_e$ still holds after line 6. An identical argument shows that $y_e \in Y_e$ still holds after 8. Finally, (6.46) implies that $\cos(\theta_e) \in COS_{\Theta_e}$ after line 9. Thus, by Theorem 13, $\theta_e \in \Theta_e$ still holds after line 10. Therefore, (6.49) holds.

Now, we verify Condition 2 of Assumption 6. Choose any $(\hat{t}, \hat{Z}, \hat{P}) \in D_{\mathcal{R}}$. By definition, $\hat{\Theta}_e \subset (-\pi, \pi)$. By openness of $(-\pi, \pi)$ and the definition of the Hausdorff metric, there must exist $\epsilon > 0$ such that $\Theta_e \subset (-\pi, \pi)$ for all $\Theta_e \in \mathbb{IR}$ satisfying $d_H(\Theta_e, \hat{\Theta}_e) < \epsilon$. By the definition of $D_{\mathcal{R}}$, this implies that $(t, Z, \hat{P}) \in D_{\mathcal{R}}$ for all $t \in B_\epsilon(\hat{t})$ and $Z \in B_\epsilon(\hat{Z})$.

Finally, we verify Condition 3 of Assumption 6 by arguing that every line of the algorithm defines its output as a locally Lipschitz continuous function of all variables on which it depends. It follows that $\mathcal{R}$ is a finite composition of locally Lipschitz functions and is therefore locally Lipschitz.

By Theorem 2.1.1 in [101], the interval operations $+$, $-$, $\times$, x^2, cos, sin, and division by a nonzero constant are all locally Lipschitz continuous on their domains. Moreover, by Theorems 10 and 14, $\bar{\sqrt{}}$ and $\bar{\arccos}$ are locally Lipschitz continuous on their domains as well. Finally, the extended intersection $\bar{\cap}$ is Lipschitz continuous by Theorem 17. Combining these facts, we conclude that each of the lines 4–10 and 12–15 defines its output by a composition of locally Lipschitz functions, and is therefore locally Lipschitz with respect to all arguments, as desired. Therefore, the entire algorithm is Lipschitz continuous with respect to (t, Z, P) on all of $\mathcal{D}_{\mathcal{R}}$, which is a stronger condition than Condition 3 of Assumption 6. $\square$

Example 2. Next, we consider trajectory tracking for the following extended model of a full size autonomous road vehicle of length $l = 2$ m [121]:

$$\begin{aligned}
\dot{x} &= v\cos\theta, \\
\dot{y} &= v\sin\theta, \\
\dot{\theta} &= v\frac{\tan\delta}{l}, \\
\dot{\delta} &= u_1, \\
\dot{v} &= \omega_2. \qquad (6.51)
\end{aligned}$$

Above, x and y are the vehicle positions, θ is the heading angle, δ is the steering angle, and v is the vehicle velocity. The control variables are the steering angle rate u_1 and the acceleration ω_2. The reference trajectory is described by x_{ref}, y_{ref}, θ_{ref}, and v_{ref}, which are computed by solving the following simplified model using the piecewise constant control inputs α_{ref} and ω_2 described in Table 6.2:

$$\begin{aligned}
\dot{x}_{ref} &= v_{ref}\cos\theta_{ref}, \\
\dot{y}_{ref} &= v_{ref}\sin\theta_{ref}, \\
\dot{\theta}_{ref} &= \alpha_{ref}, \\
\dot{v}_{ref} &= \omega_2. \qquad (6.52)
\end{aligned}$$

We apply the tracking control law from [121], which is based on the following

Table 6.2: Reference control inputs for Example 2

Time intervals (s)	[0,0.7]	[0.7,1.4]	[1.4,2.1]
α_{ref} (rad/s)	10	-4.68	1.59
ω_2 (m/s^2)	0.1	-2.05	1.16
Time intervals (s)	[2.1,2.8]	[2.8,3.5]	[3.5,4.2]
α_{ref} (rad/s)	-1.67	-2.9	4.15
ω_2 (m/s^2)	2.3	-0.05	-4.88
Time intervals (s)	[4.2,4.9]	[4.9,5.6]	[5.6,6.3]
α_{ref} (rad/s)	-0.35	-3.63	3.56
ω_2 (m/s^2)	5.47	-0.56	-2.23

global diffeomorphic coordinate transformation:

$$\begin{aligned}
e_t &= \cos(\theta_{ref})(x - x_{ref}) + \sin(\theta_{ref})(y - y_{ref}),\\
e_n &= -\sin(\theta_{ref})(x - x_{ref}) + \cos(\theta_{ref})(y - y_{ref}),\\
e_\theta &= \theta - \theta_{ref},\\
v &= v,\\
\kappa_\delta &= \tan(\delta)/l. \qquad (6.53)
\end{aligned}$$

Noting that $\dot{\kappa}_\delta = [l^{-1} + l\kappa_\delta^2]u_1$, consider the virtual control variable $\omega_1 = [l^{-1} + l\kappa_\delta^2]u_1$. The feedback law given in [121] is,

$$\begin{aligned}
\omega_1 &= -e_\theta v + \dot{\xi} - k_4(\kappa_\delta - \xi),\\
\omega_2 &= \dot{v}_{ref} - k_1 e_t - k_3(v - v_{ref}) + k_2 e_\theta^2 - e_\theta \kappa_{ref}, \qquad (6.54)
\end{aligned}$$

where $k_1 = 2$, $k_2 = 3$, $k_3 = 1$, $k_4 = 10$, and

$$\begin{aligned}
\xi &= \kappa_{ref} - k_1[e_t h_1(e_\theta) + e_n h_2(e_\theta)] - k_2 e_\theta, \qquad (6.55)\\
\dot{\xi} &= \dot{\kappa}_{ref} - k_1[\dot{e}_t h_1(e_\theta) - e_t \dot{e}_\theta (h_2(e_\theta) + h_3(e_\theta)) + \dot{e}_n h_2(e_\theta)\\
&\quad - e_n \dot{e}_\theta h_4(e_\theta)] - k_2 \dot{e}_\theta.
\end{aligned}$$

Above, h_1–h_4 are the functions defined in Definition 11 and $\kappa_{ref} = \dot{\theta}_{ref}/v_{ref}$.

The reachability problem of interest is to compute bounds on the solutions of the closed-loop system consisting of (6.53)–(6.55) subject to uncertain initial conditions described by $(e_t, e_n, e_\theta) \in [0,1]$ m $\times$ $[0,1]$ m $\times$ $[-\pi/6, \pi/6]$. We do not consider any uncertain parameters in this example. As in Example 1, the most straightforward approach to compute such bounds is to apply the standard DI method directly to (6.53) with the control law (6.54). This requires an inclusion function for the closed-loop right-hand side functions, which would be computed as follows. Given intervals X, Y, Θ, Δ, and V, interval bounds on the error states (6.53) are first computed by evaluating (6.53) in interval arithmetic. Then, interval bounds on ω_1 and ω_2 are computed by evaluating (6.54) in interval arithmetic. Next, bounds on the control input u_1 are computed from $u_1 = \omega_1/[1/l + l\kappa_\delta^2]$, and bounds on the right-hand sides of (6.53) are finally computed by interval arithmetic. As discussed in Example 1, this approach suffers from a major interval dependency problem and leads to very conservative bounds (not shown).

Following the strategy proposed in Example 1, a better approach is to directly bound the error coordinates used in the feedback law. After some simplification, the error dynamics are

$$\begin{aligned}
\dot{e}_t &= v\cos e_\theta - v_{ref}[1 - \kappa_{ref} e_n], \\
\dot{e}_n &= v\sin e_\theta - v_{ref}\kappa_{ref} e_t, \\
\dot{e}_\theta &= v\kappa_\delta - v_{ref}\kappa_{ref}, \\
\dot{\kappa}_\delta &= \omega_1, \\
\dot{v} &= \omega_2.
\end{aligned} \tag{6.56}$$

Note that the right-hand side of the ODE describing κ_δ is written in terms of ω_1 rather than u_1. This avoids having to compute u_1 using $u_1 = \omega_1/[1/l + l\kappa_\delta^2]$ and then subsequently compute $\dot{\kappa}_\delta$ using $\dot{\kappa}_\delta = [1/l + l\kappa_\delta^2]u_1$. Although the latter approach is equivalent in real arithmetic, it would result in a more conservative interval evaluation of $\dot{\kappa}_\delta$.

The results of applying standard DI to the closed-loop error system consisting of (6.56) with (6.54) and (6.55) are show in Figures 6.8–6.9 (blue). The bounds are significantly tighter those obtained by applying DI directly in the original coordinates (not shown). However, this approach still does not produce effective bounds after 0.4 s.

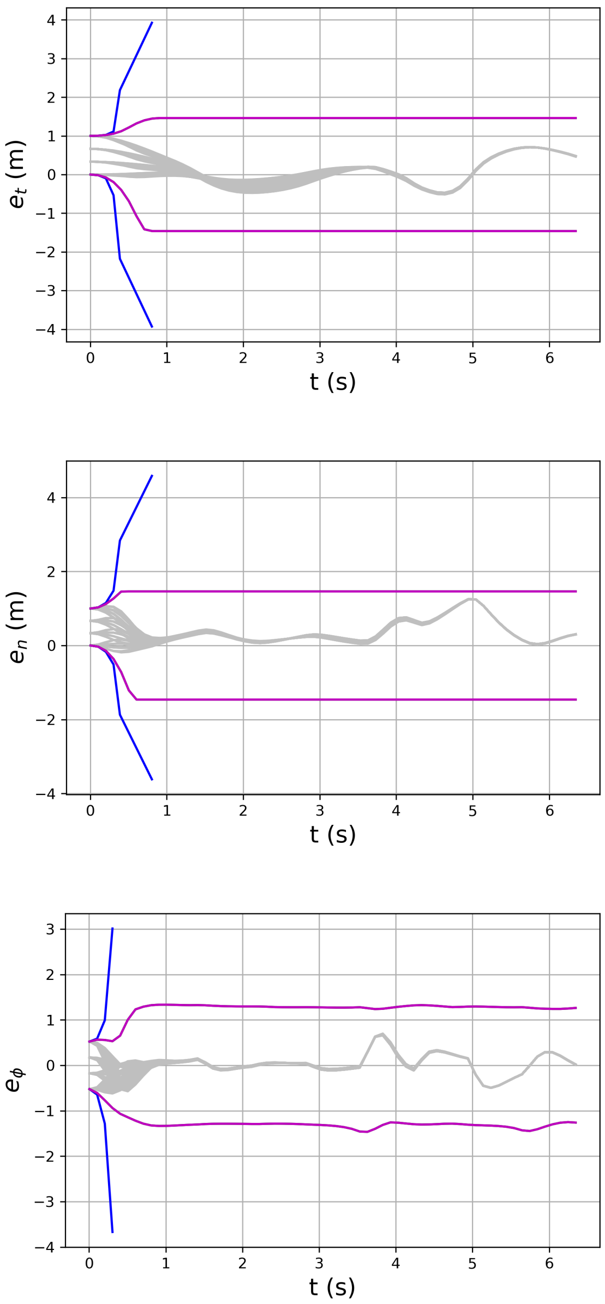

Figure 6.8: Example 2: Bounds on the error coordinates (e_t, e_n, e_ϕ) produced by (i) applying standard DI to (6.56) (blue), (ii) applying redundancy-based DI to (6.56) and (6.58) with manufactured invariants (6.57) (green), and (iii) applying redundancy-based DI to (6.56), (6.58), and (6.51) with manufactured invariants (6.57), (6.61) and (6.53) (purple) with 500 sampled trajectories (gray).

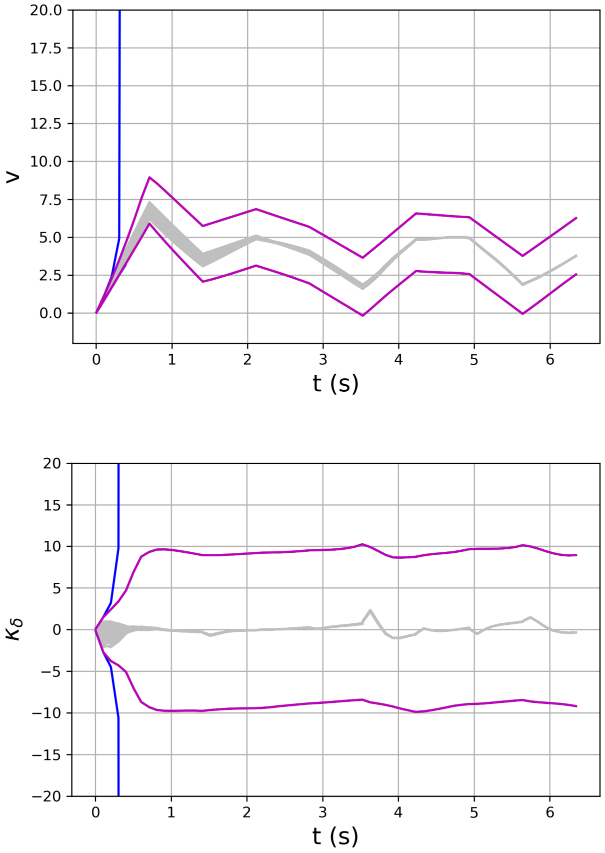

Figure 6.9: Example 2: Bounds on (v, κ_δ) produced by (i) applying standard DI to (6.56) (blue), (ii) applying redundancy-based DI to (6.56) and (6.58) with manufactured invariants (6.57) (green), and (iii) applying redundancy-based DI to (6.56), (6.58), and (6.51) with manufactured invariants (6.57), (6.61) and (6.53) (purple) with 500 sampled trajectories (gray).

To achieve further improvements, we now manufacture invariants for (6.56) following the method in [26]. As in Example 1, inspection shows that affine combinations of the states will not result in fortuitous cancellations of nonlinear and uncertain terms for this problem, as they do for most models considered in [26]. Therefore, we need to manufacture invariants using nonlinear combinations of the states. Following the same strategy used in Example 1, we consider the following two Lyapunov functions and the additional variable e_δ used in the analysis of the controller (6.54) in [121]:

$$\begin{aligned}
\mathcal{L} &= [k_1 e_t^2 + k_1 e_n^2 + e_\theta^2 + (v - v_{ref})^2]/2, \\
\mathcal{L}_C &= \mathcal{L} + (1/2)e_\delta^2, \\
e_\delta &= \kappa_\delta - \xi. \qquad (6.57)
\end{aligned}$$

To use these functions as a manufactured invariants, we define $\mathcal{L}$, $\mathcal{L}_C$, and e_δ as new state variables and augment (6.56) with the ODEs derived by differentiating them with respect to time. The ODEs obtained in this way benefits from several simplifications, ultimately leading to the forms

$$\begin{aligned}
\dot{\mathcal{L}} &= -v_{ref} k_2 e_\theta^2 - k_3 (v - v_{ref})^2, \\
\dot{\mathcal{L}}_C &= -v_{ref} k_2 e_\theta^2 - k_3 (v - v_{ref})^2 - k_4 (\kappa_\delta - \xi)^2, \\
\dot{e}_\delta &= -e_\phi v - k_4 e_\delta. \qquad (6.58)
\end{aligned}$$

Let $\mathbf{z} = (e_t, e_n, e_\theta, k_\delta, v, \mathcal{L}, \mathcal{L}_C, e_\delta)$ and $\mathbf{p} = \emptyset$ be shorthand for generic augmented state and uncertain parameter vectors. Then, by definition, the augmented system consisting of (6.56) and (6.58) satisfies Assumption 5 with the *a priori* enclosure

$$G = \Big\{(t, \mathbf{z}, \mathbf{p}) \in \mathbb{R}^9 : (6.57) \text{ holds}\Big\}. \qquad (6.59)$$

To apply the redundancy-based DI method in Corollary 5 using this G, we need to define an inclusion function $\mathcal{R}$ satisfying Assumption 6. Given any $(t, Z) \in \mathbb{R} \times \mathbb{IR}^8$ with Z denoted component-wise by $Z = E_t \times E_n \times E_\theta \times K_\delta \times V \times L \times L_C \times E_\delta$, any

$\mathbf{z} \in Z$ such that $(t, \mathbf{z}, \mathbf{p}) \in G$ must satisfy the following inclusions:

$$\begin{aligned}
e_t &\in \sqrt[-]{\frac{1}{k_1}(2L - E_\theta^2 - k_1 E_n^2 - (V - \upsilon_{ref})^2)}, \qquad (6.60)\\
e_n &\in \sqrt[-]{\frac{1}{k_1}(2L - E_\theta^2 - k_1 E_t^2 - (V - V_{ref})^2)},\\
e_\theta &\in \sqrt[-]{2L - k_1 E_t^2 - k_1 E_n^2 - (V - \upsilon_{ref})^2},\\
\upsilon &\in \sqrt[-]{2L - k_1 E_t^2 - k_1 E_n^2 - E_\theta^2} + \upsilon_{ref},\\
\mathcal{L} &\in [k_1 E_t^2 + k_1 E_n^2 + E_\theta^2 + (V - \upsilon_{ref})^2]/2,\\
\mathcal{L} &\in \mathcal{L}_C - 0.5 E_\delta^2,\\
\mathcal{L}_C &\in \mathcal{L} + 0.5 E_\delta^2,\\
e_\delta &\in 2(L_C - L),\\
e_\delta &\in K_\delta - \Xi,\\
\kappa_\delta &\in E_\delta + \Xi.
\end{aligned}$$

Our proposed definition of $\mathcal{R}$ based on these inclusions is given in Algorithm 10. The refinements based on these inclusions are done in lines 4–19, while the final enclosure of $\mathbf{f}$ (i.e., the right-hand sides of (6.56) and (6.58)) is computed in lines 21–30. All set operations are done using standard interval arithmetic or the operations defined in §6.3, and we choose the number of iterations as $l = 2$. A formal proof that this algorithm satisfies Assumption 6 is given at the end of this subsection.

Figures 6.8–6.9 show the bounds on the error states computed by applying standard DI to (6.56) (Method (i), blue) and by applying redundancy-based DI (Corollary 5) to the augmented system (6.56) and (6.58) with $\mathcal{R}$ defined by Algorithm 10 (Method (ii), green). Note that the non-smoothness of the bounding trajectories in some figures is caused by the piecewise constant inputs used to generate the reference trajectory. While the bounds computed by standard DI rapidly diverge to ±∞, the bounds computed using redundancy-based DI are much more accurate and do not diverge.

This indicates that using Lyapunov functions as manufactured invariants is very effective for this example as well.

Since the Lyapunov functions $\mathcal{L}$ and $\mathcal{L}_C$ are monotonically decreasing by (6.58), we can conclude that $[k_1 e_t^2 + k_1 e_n^2 + e_\theta^2 + (v - v_{ref})^2]/2 \leq \mathcal{L}(t = 0)$ and $[k_1 e_t^2 + k_1 e_n^2 + e_\theta^2 + (v - v_{ref})^2 + e_\delta^2]/2 \leq \mathcal{L}_C(t = 0)$. Thus, even without DI, the Lyapunov functions alone imply that the state variables e_t, e_n, e_θ, $v - v_{ref}$, and e_δ are bounded within two ellipsoids. Before 0.8 s, the bounds on e_t and e_n produced by DI (Method (ii)) are tighter than the bounds given by these ellipsoids. However, as time goes on, they gradually approach and overlap with the bounds given by the ellipsoids. In contrast, DI produces significantly better bounds than the Lyapunov functions alone for e_ϕ. Specifically, the ellipsoids computed by the Lyapunov functions indicate that the absolute value of e_ϕ should be bounded by 2.07. This is much larger than the bounds computed by the Method (ii), which converges around 1.28. Hence, the combination of DI with the Lyapunov functions as additional variables in Method (ii) achieves tighter bounds than can be inferred from either DI or the Lyapunov functions alone.

Given bounds on the error coordinates, bounds on the vehicle's position in the original coordinates can be obtained by evaluating the following inverse coordinate transformation in interval arithmetic:

$$\begin{aligned}
x &= \cos(\theta_{ref}) e_t - \sin(\theta_{ref}) e_n + x_{ref} \qquad (6.61)\\
y &= \sin(\theta_{ref}) e_t + \cos(\theta_{ref}) e_n + y_{ref}\\
\theta &= e_\theta + \theta_{ref}.
\end{aligned}$$

Figure 6.10 shows the resulting bounds for each method. It can be seen that the accuracy of the redundancy-based DI method is retained in the original coordinates. As in Example 1, these bounds certainly leave room for improvement, but do appear to be accurate enough to support some motion planning and collision avoidance tasks.

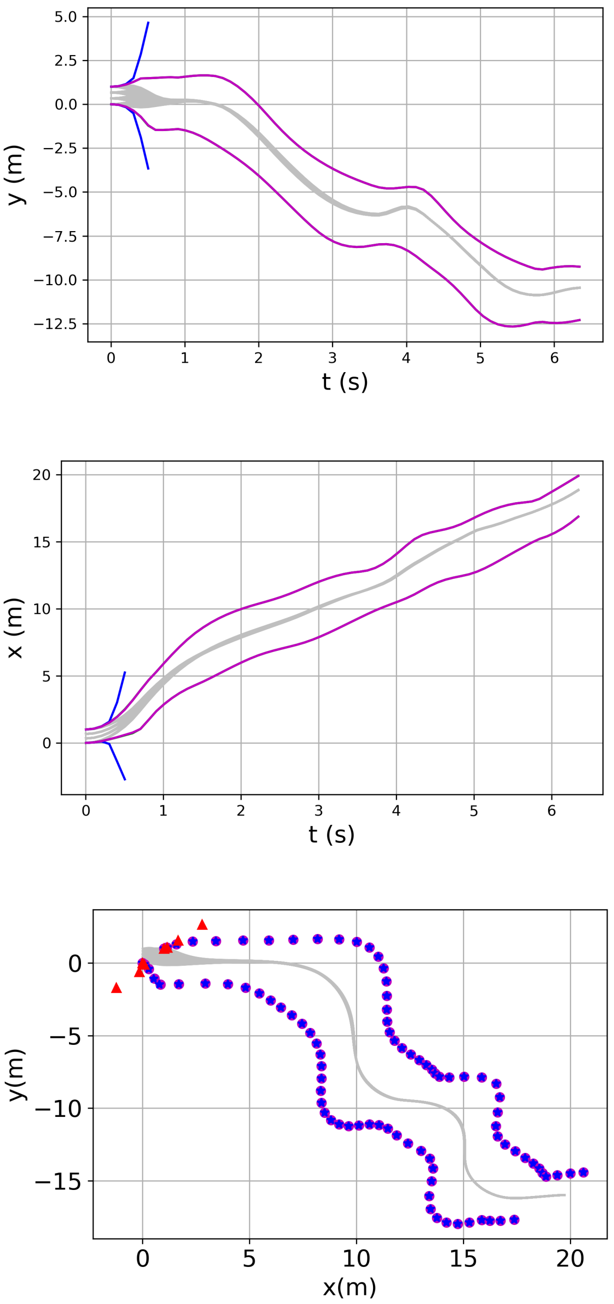

Figure 6.10: Example 2: Bounds on the original coordinates (x, y) produced by (i) applying standard DI to (6.56) (blue), (ii) applying redundancy-based DI to (6.56) and (6.58) with manufactured invariants (6.57) (green), and (iii) applying redundancy-based DI to (6.56), (6.58), and (6.51) with manufactured invariants (6.57), (6.61) and (6.53) (purple) with 500 sampled trajectories (gray).

To make further improvements, we again consider using the original and error coordinates simultaneously with the coordinate transformation serving as additional manufactured invariants. Specifically, we now augment (6.56) and (6.58) with (6.51). In addition to (6.57), the states of this augmented system also satisfy the invariants (6.53) and (6.61). Therefore, we can define the following *a priori* enclosure with $\mathbf{z} = (e_t, e_n, e_\theta, k_\delta, v, \mathcal{L}, \mathcal{L}_C, e_\delta, x, y, \theta, \delta)$:

$$G = \Big\{(t, \mathbf{z}, \mathbf{p}) \in \mathbb{R}^{13} : (6.57), (6.53), \text{and } (6.61) \text{ hold}\Big\}. \tag{6.62}$$

To apply the redundancy-based DI method in Corollary 5 using this G, we again define an inclusion function $\mathcal{R}$ by including additional refinements based on (6.53) and (6.61).

Given any $(t, Z) \in \mathbb{R} \times \mathbb{IR}^{13}$ with Z denoted component-wise by $Z = E_t \times E_n \times E_\theta \times V \times L \times L_C \times E_\delta \times K_\delta \times X \times Y \times \Theta \times \Delta$, any $\mathbf{z} \in Z$ such that $(t, \mathbf{z}, \mathbf{p}) \in G$ must satisfy:

$$\begin{aligned}
e_t &\in \cos(\theta_{ref})(X - x_{ref}) + \sin(\theta_{ref})(Y - y_{ref}), \qquad (6.63)\\
e_n &\in -\sin(\theta_{ref})(X - x_{ref}) + \cos(\theta_{ref})(Y - y_{ref}),\\
e_\theta &\in \Theta - \theta_{ref},\\
\kappa_\delta &\in \tan(\Delta)/l,\\
x &\in \cos(\theta_{ref})E_t - \sin(\theta_{ref})E_n + x_{ref},\\
y &\in \sin(\theta_{ref})E_t + \cos(\theta_{ref})E_n + y_{ref},\\
\theta &\in E_\theta + \theta_{ref},\\
\delta &\in \arctan(lK_\delta).
\end{aligned}$$

Our proposed definition of $\mathcal{R}$ based on these refinements is given in Algorithm 11. Compared to Algorithm 10, Algorithm 11 adds the refinements (6.63) in lines 5–12 and computes bounds on the right-hand sides of the ODEs (6.51) (which are now included in the augmented system) in lines 15–18.

Algorithm 11 An implementation of $\mathcal{R}$

```
 1: function $\mathcal{R}(t, Z, P)$
 2:     $(E_t, E_n, E_\theta, K_\delta, V, L, L_C, E_\delta, X, Y, \Theta, \Delta) \leftarrow Z$
 3:     for $i = 1$ to $l$ do
 4:         Apply lines 4–19 in Algorithm 10
 5:         $X = X \bar{\cap} (\cos(\theta_{ref})E_t - \sin(\theta_{ref})E_n + x_{ref})$
 6:         $Y = Y \bar{\cap} (\sin(\theta_{ref})E_t + \cos(\theta_{ref})E_n + y_{ref})$
 7:         $\Theta = \Theta \bar{\cap} (E_\theta + \theta_{ref})$
 8:         $\Delta = \Delta \bar{\cap} \arctan(lK_\delta)$
 9:         $E_t = E_t \bar{\cap} (\cos(\theta_{ref})(X - x_{ref}) + \sin(\theta_{ref})(Y - y_{ref}))$
10:         $E_n = E_n \bar{\cap} (-\sin(\theta_{ref})(X - x_{ref}) + \cos(\theta_{ref})(Y - y_{ref}))$
11:         $E_\theta = E_\theta \bar{\cap} (\Theta - \theta_{ref})$
12:         $K_\delta = K_\delta \bar{\cap} \frac{\tan(\Delta)}{l}$
13:     end for
14:     Apply lines 21–30 in Algorithm 10
15:     $\Sigma_9 \leftarrow V \cos\Theta$
16:     $\Sigma_{10} \leftarrow V \sin\Theta$
17:     $\Sigma_{11} \leftarrow V \frac{\tan\Delta}{l}$
18:     $\Sigma_{12} \leftarrow \Sigma_4/(1/l + lK_\delta^2)$
19: return $\Sigma \leftarrow (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \Sigma_7, \Sigma_8, \Sigma_9, \Sigma_{10}, \Sigma_{11}, \Sigma_{12})$
20: end function
```

The results applying redundancy-based DI (Corollary 5) with $\mathcal{R}$ defined by Algorithm 11 (Method (iii), purple) are shown in Figures 6.8–6.10. In both original and error coordinates, the bounds from this method lie entirely behind the plotted bounds for Method (ii), indicating that the model redundancy offered by using both the original and error coordinates simultaneously is ineffective at mitigating the dependency problem for this example, as it was for Example 1.

In terms of computational cost, standard DI is the most efficient method tested with a cost of 0.0018 s. However, this time is misleading because integration was stopped early due to divergence of the bounds. Method (ii) using the Lyapunov functions requires 0.222 s, and Method (iii) using the Lyapunov function and the coordinate transformations requires 1.83 s for a time horizon of 6 s. For comparison, approximating the reachable set by simulating solutions on a grid with 20 points for every uncertain initial condition (i.e., 8000 trajectories) requires 14.2 s. Method (ii) clearly offers the best trade-off between accuracy and efficiency, producing effective bounds with a computational time that is equivalent to sampling about 125 real trajectories, and that is roughly 27× faster than the real travel time for this vehicle.

We close this subsection by proving that Algorithm 10 satisfies Assumption 6. The proof for Algorithm 11 is a straightforward extension of the same arguments and is omitted for brevity. Consider the augmented system consisting of (6.56) and (6.58) and let $\mathbf{z} \equiv (e_t, e_n, e_\theta, k_\delta, v, \mathcal{L}, \mathcal{L}_C, e_\delta)$ and $\mathbf{p} = \emptyset$ denote generic state and parameter vectors. Similarly, let $Z = (X_e, Y_e, \Theta_e, V)$ and $P = \emptyset$ denote generic state and parameter interval vectors.

Theorem 19. *Define* $D_{\mathcal{R}} \equiv \{(t, Z, P) \in \mathbb{R} \times \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_p} : E_\theta \subset (-\pi/2, \pi/2)$. *Algorithm 10 is well defined for every* $(t, Z, P) \in D_{\mathcal{R}}$. *Moreover, Assumption 6 holds with* $\mathcal{R}(t, Z, P)$ *defined by Algorithm 10.*

Proof. Choose any $(t, Z, P) \in D_{\mathcal{R}}$. Algorithm 10 is well defined for (t, Z, P) if and only if no domain violations occur when evaluating the inclusion functions in lines 4–19 and 21–30. By the extended intersection in lines 4–19, lines of inclusion functions 21–30 are always evaluated in $[t, t], Z', P$ with $Z' \subset Z$. Thus, we have $E_\theta \subset (\pi/2, \pi/2)$ always evaluated in the domain of H_1–H_4. Therefore, there are no domain violations in lines 17 and 24. By the extended intersection in lines 7, 9, 11, 13, and 15, we have SQ_{E_δ}, SQ_{E_θ}, SQ_{E_t}, SQ_{E_n} and SQ_{E_v} are always lying in $\mathbb{IR}^+$, which is the appropriate domain of the extended square root function in lines 8, 10, 12, 14, and 16. Since lines

4, 5, 6, 7, 9, 11, 13, 15, 18, and 19 always lie in the appropriate domains, $\mathcal{R}$ is always well defined in $D_{\mathcal{R}}$.

To verify Condition 1 of Assumption 6, choose any $(t, Z, P) \in D_{\mathcal{R}}$, and let $\Sigma = \mathcal{R}(t, P, Z)$ be the output of Algorithm 10. We must show that

$$\Sigma \supset \{\mathbf{f}(t, \mathbf{z}, \mathbf{p}) : (\mathbf{z}, \mathbf{p}) \in Z \times P,\ (t, \mathbf{z}, \mathbf{p}) \in G\}, \tag{6.64}$$

where $\mathbf{f}$ denotes the right-hand side functions of (6.56) and (6.58) and

$$G = \left\{(t, \mathbf{z}) \in \mathbb{R}^9 : (6.57),\ (6.56),\ \text{ and }\ (6.61)\ \text{hold}\right\}. \tag{6.65}$$

Choose any $(\mathbf{z}, \mathbf{p}) \in Z \times P$ satisfying $(t, \mathbf{z}, \mathbf{p}) \in G$ and let $(e_t, e_n, e_\theta, k_\delta, v, \mathcal{L}, \mathcal{L}_C, e_\delta) = \mathbf{z}$. Since $\mathbf{z} \in Z$, the following inclusion all hold immediately after line 2: $e_t \in E_t$, $e_n \in E_n$, $e_\theta \in E_\theta$, $k_\delta \in K_\delta$, $v \in V$, $\mathcal{L} \in L$, $\mathcal{L}_C \in L_C$, $e_\delta \in E_\delta$. If these inclusions remain valid when line 21 is reached, then (6.64) must hold because lines 21–30 are direct interval evaluations of the right-hand side functions of (6.56) and (6.58) (i.e., $\mathbf{f}$). Thus, it suffices to show that these inclusions are maintained through lines 4–19. By (6.60) and the extended intersection, $\mathbf{z}$ satisfies the inclusions $\mathcal{L}_C \in L_C$ after line 4, $\mathcal{L} \in L$ after lines 5 and 6, $e_\delta^2 \in SQ_{E_\delta}$, $e_\theta^2 \in SQ_{E_\theta}$, $e_t^2 \in SQ_{E_t}$, $e_n^2 \in SQ_{E_n}$, $(v - v_{ref})^2 \in SQ_{E_v}$ after lines 7, 9, 11, 13, and 15, $e_\delta^2 \in E_\delta$ after line 18, $\kappa_\delta \in K_\delta$ after line 19. By Theorem 9, we have $e_\delta \in E_\delta$, $e_\theta \in E_\theta$, $e_t \in E_t$, $e_n \in E_n$, $V - v_{ref} \in E_v$ in lines 8, 10, 12, 14 and 16. By the definition of extended intersection, (6.64) holds after line 19. Therefore, (6.64) holds for the entire algorithm.

Now, we verify Condition 2 of Assumption 6. Choose $(\hat{t}, \hat{Z}, \hat{P}) \in D_{\mathcal{R}}$. By definition, $\hat{E}_\Theta \subset (-\pi/2, \pi/2)$. By openness of $(-\pi/2, \pi/2)$ and the definition of the Hausdorff metric, there must exist $\epsilon > 0$ such that $E_\Theta \subset (-\pi, \pi)$ for all $E_\Theta \in \mathbb{IR}$ satisfying $d_H(E_\Theta, \hat{E}_\Theta) < \epsilon$. By the definition of $D_{\mathcal{R}}$, this implies that $(t, Z, \hat{P}) \in D_{\mathcal{R}}$ for all $t \in B_\epsilon(\hat{t})$ and $Z \in B_\epsilon(\hat{Z})$.

Finally, we verify Condition 3 of Assumption 6 by arguing that every line of the algorithm defines its output as a locally Lipschitz continuous function of all variables on which it depends. It follows that $\mathcal{R}$ is a finite composition of locally Lipschitz functions and is therefore locally Lipschitz continuous.

By Theorem 2.1.1 in [101], the interval operations $+$, $-$, $\times$, x^2, cos, sin, and division by a nonzero constant are all locally Lipschitz continuous on their domains. Moreover, by Theorems 10 and 16, $\bar{\sqrt{}}$ and H_1–H_4 are locally Lipschitz continuous on their domains as well. Finally, the extended intersection $\bar{\cap}$ is Lipschitz continuous by Theorem 17. Combining these facts, we conclude that each of the lines 4–19 and 21–30 defines its output by a composition of locally Lipschitz functions, and is therefore locally Lipschitz with respect to all arguments, as desired. Therefore, the entire algorithm is Lipschitz continuous with respect to (t, Z, P) on all of $\mathcal{D}_{\mathcal{R}}$, which is a stronger condition than Condition 3 of Assumption 6. $\square$

Example 3. Consider the same vehicle dynamics as in (6.37). The velocity is considered to be a time-invariant uncertain parameter satisfying $v \in V$ for some interval V. The initial condition may also be uncertain with $\mathbf{x}_0 \in X_0$. The control objective is to manipulate ω to track a smooth path C. Specifically, $C \equiv \{(x_{ref}(s), y_{ref}(s)) : s \in [0, \bar{s}]\}$ where $(x_{ref}, y_{ref}) \in \mathcal{C}^2([0, \bar{s}], \mathbb{R}^2)$ and $\bar{s}$ is the arc length of C.

We consider the path tracking controller proposed in [122], which is based on the following curvilinear coordinate transformation. First, the *curvature* of C at $(x, y) \in C$ is defined as the derivative of the unit tangent at (x, y) with respect to the arc length of the path. Let $c : [0, \bar{s}] \to \mathbb{R}$ map each $s \in [0, \bar{s}]$ to the curvature of C at the point $(x_{ref}(s), y_{ref}(s))$ and define $\bar{c} \equiv \min_{s \in [0, \bar{s}]} |c(s)|$.

Assume that, at any point $(x, y) \in C$, 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 [122]. Consider a single trajectory $(x(t), y(t), \theta(t))$ of (6.37) corresponding to some $\mathbf{x} \in X_0$, $v \in V$, and control input $\omega : [t_0, t_f] \to \mathbb{R}$. Assume that the minimum distance between the position

$(x(t), y(t))$ and the path C remains less than $1/\bar{c}$ for all $t \in [t_0, t_f]$. In this case, the projection of $(x(t), y(t))$ onto C is well-defined and we may define the curviliear coordinate $s : [t_0, t_f] \to \mathbb{R}$ for this trajectory as

$$s(t) = \underset{\gamma \in [0, \bar{s}]}{\operatorname{argmin}} \|((x(t), y(t)) - (x_{ref}(\gamma), y_{ref}(\gamma)))\|_2, \tag{6.66}$$

Following [67], define the x and y coordinate errors by $d_x(t) \equiv x(t) - x_{ref}(s(t))$ and $d_y(t) \equiv y(t) - y_{ref}(s(t))$. Moreover, define the unit tangent to C at the point $(x_{ref}(s(t)), y_{ref}(s(t)))$ by

$$\mathbf{n}(t) = (n_x(t), n_y(t)) \equiv \frac{\left(\frac{\partial x_{ref}}{\partial s}(s(t)), \frac{\partial y_{ref}}{\partial s}(s(t))\right)}{\left\|\left(\frac{\partial x_{ref}}{\partial s}(s(t)), \frac{\partial y_{ref}}{\partial s}(s(t))\right)\right\|}. \tag{6.67}$$

Define the tracking error $e : [t_0, t_f] \to \mathbb{R}$ by

$$e(t) = d_x(t) n_y(t) - d_y(t) n_x(t). \tag{6.68}$$

The tracking error is positive if the vehicle is to the right of the curve C and negative if it is to the left. Finally, define the tracking angle error $\theta_e : [t_0, t_f] \to \mathbb{R}$ as the difference between the heading angle $\theta(t)$ and angle of the tangent vector $\mathbf{n}(t)$:

$$\theta_e(t) = \theta(t) - \arctan 2\left(\frac{\partial x_{ref}}{\partial s}(s(t)), \frac{\partial y_{ref}}{\partial s}(s(t))\right). \tag{6.69}$$

According to [122], the trajectory $(s(t), e(t), \theta_e(t))$ defined in this way satisfies the following system of ODEs:

$$\begin{aligned}\dot{s} &= \frac{v\cos(\theta_e)}{1-c(s)e},\\ \dot{e} &= v\sin(\theta_e),\\ \dot{\theta}_e &= \omega - \frac{vc(s)\cos(\theta_e)}{1-c(s)e}.\end{aligned} \tag{6.70}$$

Here, we have s, e, θ_e uniquely defined if $|e| < \left|\frac{1}{\bar{c}}\right|$, where $\bar{c} = c(s) = 1/30$. Thus, the domain of e is $e \in (-30, 30)$. Following [122], we apply the following tracking control law in these coordinates:

$$\omega = \frac{vc(s)\cos(\theta_e)}{1-c(s)e} - g_1\theta_e - (g_2 v h_2(\theta_e))\, e, \tag{6.71}$$

where h_2 is defined in Definition 11 and the gains are $g_1 = 5.71\sqrt{v^2+0.1}$ and $g_2 = 4$. This closed-loop system is proven to be asymptotically stable in [122] for systems with certain velocity. However, for safety verification it is of interest to know how far the vehicle can deviate from the desired path C under uncertainty. Therefore, our aim is to compute bounds on the solutions of the closed-loop system consisting of (6.70) and (6.71) and bound the position of the vehicle at each instant in time.

Let the time-invariant uncertain parameter $v \in [5, 6]$ m/s^2 and the initial condition $\mathbf{x}_0 = (0, 1, \pi/6)$. As in Example 1, a straight forward way is to compute bounds on the error coordinates (6.70) with feedback law (6.71) using standard DI. This requires an inclusion function for the closed-loop right-hand side functions computed as follows. Given intervals S, E, and Θ_e, interval bounds on ω is computed by evaluating (6.71). Then, the bounds on the right-hand side of (6.70) are evaluated using interval arithmetic. As discussed in Examples 1 and 2, this method leads to conservative bounds because of the interval dependency problem caused by interval arithmetic evaluating ω and the right-hand side of (6.70).

Following similar strategies in Example 1, the right hand side function of $\dot{\theta}_e$ can be simplified by plugging in (6.71):

$$\dot{\theta}_e = -g_1\theta_e - g_2 v h_2(\theta_e)e.$$

A nice cancellation on the term $\frac{vc(s)\cos(\theta_e)}{1-c(s)e}$ reduces the dependency problem. Of course, valid bounds can be obtained by directly applying the standard DI method to this system. However, this still results very weak bounds as discussed in Examples 1 and 2. In Figure 6.11, Method (i) computes bounds of the states using the standard DI method on this system. The bounds start to diverge since 0.4 s.

To improve these bounds using the redundancy-based DI method described in Corollary 5, we need construct a valid constraint set G by introducing redundant states and ODEs. Since there is no obvious linear invariants in this example, we follow the proposed strategy in Example 1 to manufacture invariants based on Lyapunov function in [122]:

$$\mathcal{L} = \frac{1}{2}(e^2 + (1/g_2)\theta_e^2). \tag{6.72}$$

Then, augment (6.70) with

$$\dot{\mathcal{L}} = -\frac{g_1}{g_2}\theta_e^2. \tag{6.73}$$

Define $\mathbf{z} \equiv (s, e, \theta_e, \mathcal{L})$ and the uncertain parameter $p = v$. Then, the augmented system consisting of (6.70) and (6.73) satisfies Assumption 5 with the *a priori* enclosure

$$G = \left\{(t, \mathbf{z}, p) \in \mathbb{R}^6 : \mathcal{L} = \frac{1}{2}(e^2 + (1/g_2)\theta_e^2)\right\}. \tag{6.74}$$

To apply the redundancy-based DI method in Corollary 5 using this G, we follow the same procedure as in Example 1 to define an inclusion function $\mathcal{R}$ satisfying 6. Given any $(t, Z, P) \in \mathbb{R} \times \mathbb{IR}^4 \times \mathbb{IR}$ and any $\mathbf{z} = (s, e, \theta_e, \mathcal{L}) \in Z$ and $p = v \in P$ such that $(t, \mathbf{z}, p) \in G$, the following rearrangements of (6.43) must hold:

$$e^2 = 2\mathcal{L} - (1/g_2)\theta_e^2, \tag{6.75}$$

$$\theta_e^2 = g_2(2\mathcal{L} - e^2). \tag{6.76}$$

Therefore, denoting Z component-wise by $Z = S \times E \times \Theta_e \times L$, $\mathbf{z}$ must satisfy the following inclusions, where the right-hand-side is evaluated in interval arithmetic using the inclusion functions $\sqrt[-]{}$ defined in §6.3:

$$\begin{aligned} \mathcal{L} &\in \frac{1}{2}(E^2 + (1/g_2)\Theta_e^2), \\ e &\in \sqrt[-]{2L - (1/g_2)\Theta_e^2}, \\ \theta_e &\in \sqrt[-]{g_2(2L - E^2)}. \end{aligned} \tag{6.77}$$

Based on the refinement (6.77), we have the refinement algorithm defined in Algorithm 12. The inputs of the algorithm are the interval bounds of the states Z and the interval bounds of the uncertainties P. Specifically, lines 4–8 refines intervals S, E, Θ_e, and L based on the rearrangements in (6.77). Finally, the enclosures of the right-hand-side functions in (6.70) and (6.73) are computed by lines 10–13.

Algorithm 12 An implementation of $\mathcal{R}$ for Example 3

```
function $\mathcal{R}(t, Z, P)$
    $(S \times E \times \Theta_e \times L) \leftarrow Z$ and $V \leftarrow P$
    for $i = 1$ to $l$ do
        $L \leftarrow L \bar{\cap} [\frac{1}{2}(E^2 + (1/g_2)\Theta_e^2)]$
        $SQ_E \leftarrow E^2 \bar{\cap} (2L - \frac{1}{g_2}\Theta_e^2)$
        $E \leftarrow E \bar{\cap} \sqrt[-]{SQ_E}$
        $SQ_{\Theta_e} \leftarrow \Theta_e^2 \bar{\cap} (g_2(2L - E^2))$
        $\Theta_e \leftarrow \Theta_e \bar{\cap} \sqrt[-]{SQ_{\Theta_e}}$
    end for
    $\Sigma_1 \leftarrow \frac{V\cos(\Theta_e)}{1-\bar{c}E}$
    $\Sigma_2 \leftarrow V\sin(\Theta_e)$
    $\Sigma_3 \leftarrow -5.71\sqrt{V^2 + 0.1}\Theta_e - g_2 V H_2(\Theta_e)E,$
    $\Sigma_4 \leftarrow -\frac{5.71\sqrt{V^2+0.1}}{g_2} SQ_{\Theta_e}$
return $\Sigma \leftarrow (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4)$
end function
```

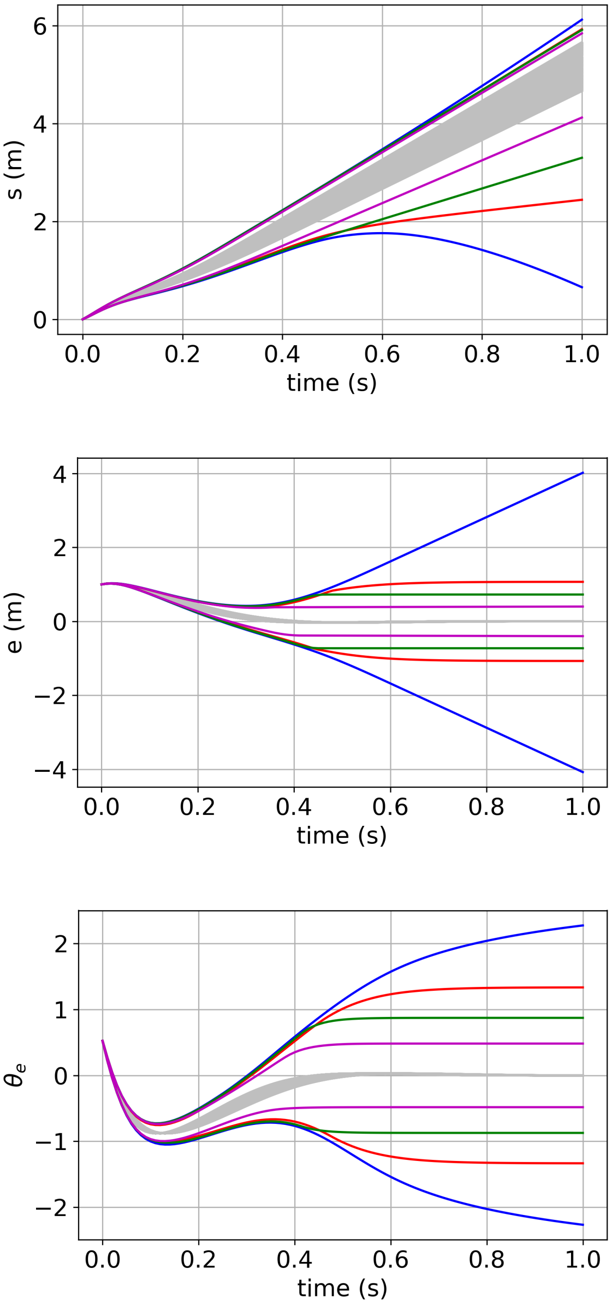

Figure 6.11: Example 3: Bounds on the error coordinates (s, e, θ_e) produced by (i) applying standard DI to (6.70) (blue), (ii) applying redundancy-based DI to (6.70) and (6.73) with manufactured invariant (6.72) (green) using Algorithm 12, (iii) applying redundancy-based DI to (6.70) and (6.73) with manufactured invariant (6.72) using the κ-operator in [117] (red), (iv) applying redundancy-based DI to (6.78) with manufactured invariants (6.79) and (6.72) using Algorithm 13 (purple) with 500 sampled trajectories (grey).

Figure 6.11 compares bounds of the states using the proposed methods. Method (ii) and (iii) compute bounds for (6.70) using redundancy-based DI by applying κ-operator in [96] and Algorithm 12 respectively as the refinement operator $\mathcal{R}$. Applying the Lyapunov invariant (6.72), the redundancy-based DI methods improve the performance to the standard DI method (i). The bounds of Method (ii) and Method (iii) converges after 0.4 s. Furthermore, applying the proposed refinement algorithm 12 based on algebraic rearrangement leads to much tighter bounds than Method (ii) using κ-operator in [96].

Now, we propose a new strategy to manufacture additional invariant constraints in this specific example. We define $\phi_1 = v\cos(\theta_e)$ and $\phi_2 = v\sin(\theta_e)$. Then, replace $v\cos(\theta_e)$ and $v\sin(\theta_e)$ in (6.70) with ϕ_1, ϕ_2. Since velocity v in (6.70) is time-invariant, the augmented system dynamics becomes:

$$\begin{aligned}
\dot{s} &= \frac{\phi_1}{1 - c(s)e}, \\
\dot{e} &= \phi_2, \\
\dot{\theta}_e &= -g_1\theta_e - g_2 e v h_2(\theta_e), \\
\dot{\mathcal{L}} &= -\frac{g_1}{g_2}\theta_e^2, \\
\dot{\phi}_1 &= -v\sin(\theta_e)\dot{\theta}_e, \\
\dot{\phi}_2 &= v\cos(\theta_e)\dot{\theta}_e,
\end{aligned} \tag{6.78}$$

where the system satisfies the constraints (6.72) and

$$\begin{aligned}
v^2 &= \phi_1^2 + \phi_2^2, \\
\phi_1 &= v\cos(\theta_e), \\
\phi_2 &= v\sin(\theta_e).
\end{aligned} \tag{6.79}$$

Define $\mathbf{z} \equiv (s, e, \theta_e, \mathcal{L}, \phi_1, \phi_2)$. Now, we define the following *a priori* enclosure with uncertain parameter $p = v$:

$$G = \left\{(t, \mathbf{z}, \mathbf{p}) \in \mathbb{R}^8 : (6.79) \text{ and } (6.72) \text{ hold}\right\}. \tag{6.80}$$

To apply the redundancy-based DI method in Corollary 5 using this G, we again define an inclusion function $\mathcal{R}$ by including additional refinements based on (6.79). Given any $(t, Z, P) \in \mathbb{R} \times \mathbb{IR}^6 \times \mathbb{IR}$ and denoting Z component-wise by $Z = S \times E \times \Theta_e \times L \times \Phi_1 \times \Phi_2$, (6.79) implies that any $\mathbf{z} \in Z$ and $p \in P$ satisfying $(t, \mathbf{z}, \mathbf{p}) \in G$ must also satisfy the following inclusions:

$$\begin{aligned}
\phi_1 &\in \cos(\Theta_e)V, \qquad (6.81)\\
\phi_2 &\in \sin(\Theta_e)V,\\
v &\in \sqrt[-]{\Phi_1^2 + \Phi_2^2},\\
\phi_1 &\in \sqrt[-]{V^2 - \Phi_2^2},\\
\phi_2 &\in \sqrt[-]{V^2 - \Phi_1^2},\\
\theta_e &\in \mathrm{ar\bar{c}cos}(\Phi_1/V),\\
\theta_e &\in \mathrm{ar\bar{c}sin}(\Phi_2/V),\\
\theta_e &\in \arctan(\Phi_2/\Phi_1),
\end{aligned}$$

where $\sqrt[-]{\ }$, $\mathrm{ar\bar{c}sin}$, and $\mathrm{ar\bar{c}cos}$ defined in §6.3 are used.

Our proposed definition of $\mathcal{R}$ based on these inclusions is given in Algorithm 13. The refinements based on these inclusions are in lines 4–18. The enclosure of $\mathbf{f}$ (i.e., the right-hand sides of (6.78)) is computed in lines 20–25. All set operations are done using standard interval arithmetic or the operations defined in §6.3, and we choose the number of iterations as $l = 2$. A formal proof that this algorithm satisfies Assumption 6 is given at the end of this subsection.

Algorithm 13 An implementation of $\mathcal{R}$ for Example 3

```
function $\mathcal{R}(t, Z, P)$
    $(S \times E \times \Theta_e \times L \times \Phi_1 \times \Phi_2) \leftarrow Z$ and $V \leftarrow P$
    for $i = 1$ to $l$ do
        Apply lines 4–8 in Algorithm 12
        $\Phi_1 \leftarrow V \cos(\Theta_e) \bar{\cap} \Phi_1$
        $\Phi_2 \leftarrow \Phi_2 \bar{\cap} V \sin(\Theta_e)$
        $SQ_V \leftarrow V^2 \bar{\cap} (\Phi_1^2 + \Phi_2^2)$
        $V \leftarrow V \bar{\cap} \bar{\sqrt{SQ_V}}$
        $SQ_{\Phi_1} \leftarrow \Phi_1^2 \bar{\cap} (V^2 - \Phi_2^2)$
        $\Phi_1 \leftarrow \Phi_1 \bar{\cap} \bar{\sqrt{SQ_{\Phi_1}}}$
        $SQ_{\Phi_2} \leftarrow \Phi_2^2 \bar{\cap} (V^2 - \Phi_1^2)$
        $\Phi_2 \leftarrow \Phi_2 \bar{\cap} \bar{\sqrt{SQ_{\Phi_2}}}$
        $COS_{\Theta_e} \leftarrow \cos \Theta_e \bar{\cap} (\Phi_1 / V)$
        $\Theta_e \leftarrow \Theta_e \bar{\cap} \mathrm{arc\bar{c}cos}(COS_{\Theta_e})$
        $SIN_{\Theta_e} \leftarrow \sin \Theta_e \bar{\cap} (\Phi_2 / V)$
        $\Theta_e \leftarrow \Theta_e \bar{\cap} \mathrm{ar\bar{c}sin}(SIN_{\Theta_e})$
        $TAN_{\Theta_e} \leftarrow \tan \Theta_e \bar{\cap} (\Phi_2 / \Phi_1)$
        $\Theta_e \leftarrow \Theta_e \bar{\cap} \arctan(TAN_{\Theta_e})$
    end for
    $\Sigma_1 \leftarrow \frac{\Phi_1}{1 - \bar{c}E}$
    $\Sigma_2 \leftarrow \Phi_2$
    $\Sigma_3 \leftarrow -\sqrt{V^2 + 0.1}\Theta_e - g_2 V H_2(\Theta_e) E,$
    $\Sigma_4 \leftarrow -\frac{\sqrt{V^2+0.1}}{g_2} SQ_{\Theta_e}$
    $\Sigma_5 \leftarrow -V \sin(\Theta_e) \Sigma_3$
    $\Sigma_6 \leftarrow V \cos(\Theta_e) \Sigma_3$
return $\Sigma \leftarrow (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6)$
end function
```

Method (iv) computes bounds on (6.78) with invariants (6.79) using refinement Algorithm 13. Figure 6.11 shows that applying both Lyapunov function and coordinate transformation using ϕ_1 and ϕ_2 as invariants in (6.79), Method (iv) computes tightest bounds on (6.78) among all compared methods. Moreover, the Lyapunov function (6.72) directly implies that the system states are always staying in the ellipsoid $\frac{1}{2}(e^2 + (1/g_2)\theta_e^2) \leq V(e(t = 0), \theta_e(t = 0))$. Thus, the absolute values of e and θ_e are approximately bounded by 1.03 (m) and 2.07 which are much larger than the bounds computed by the proposed Methods (ii)–(iv). By only applying the Lyapunov function without DI methods, the bounds of e and θ_e can be convergent. However, these bounds are very large and may not be useful for safety verification.

To transform the coordinates to the system original coordinates, an easy way is to augment (6.70) and (6.78) with the original dynamics given by [122]:

$$\begin{aligned} \dot{x} &= v\cos(\theta_d + \theta_e), \\ \dot{y} &= v\sin(\theta_d + \theta_e), \\ \dot{\theta}_d &= c(s)\dot{s}. \end{aligned} \tag{6.82}$$

Then, we apply Methods (i)–(iv) to compute bounds on the augmented systems. The inclusion functions of (6.82) can be computed by interval arithmetic by plugging in $\dot{s}$ from (6.70) or (6.78). The results given in Figure 6.12 shows that this approach does not give convergent bounds on vehicle positions, although Method (iv) produces tight and convergent bounds for the error coordinator system.

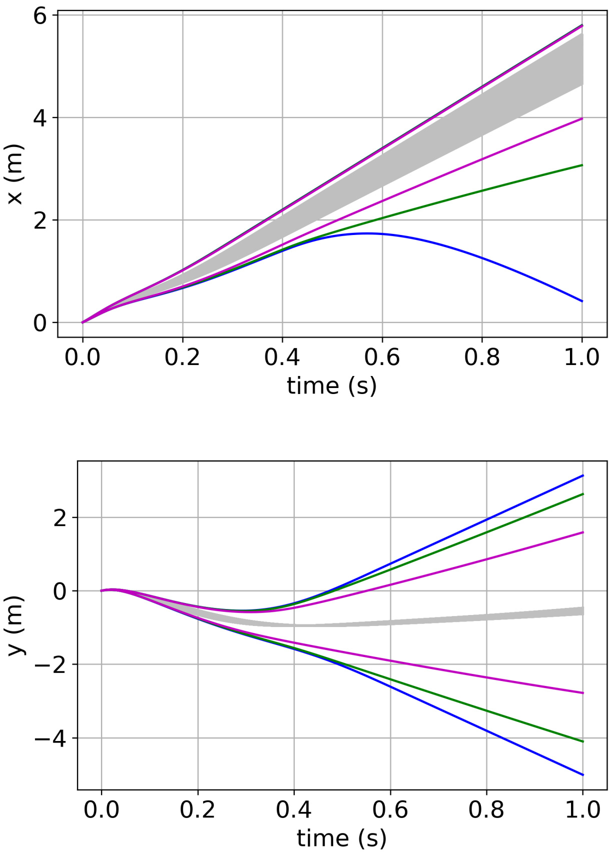

Figure 6.12: Example 3: Bounds on the original coordinates (x, y) produced by (i) applying standard DI to (6.70) (blue), (ii) applying redundancy-based DI to (6.70) and (6.73) with manufactured invariant (6.72) (green) using Algorithm 12, (iii) applying redundancy-based DI to (6.70) and (6.73) with manufactured invariant (6.72) using the κ-operator in [117] (red), (iv) applying redundancy-based DI to (6.78) with manufactured invariants (6.79) and (6.72) using Algorithm 13 (purple) with 500 sampled trajectories (grey).

Now, we introduce a method to directly compute bounds on its original coordinates using the interval bounds of e and θ_e and reference positions x_{ref} and y_{ref}. Since $e \in (-\min(|c(s))|, |\min(|c(s))|)$, we have $\dot{s} = \frac{ds}{dt} > 0$. Then, we multiply $\frac{dt}{ds}$ at both sides of (6.82). The dynamics of the reference trajectory can be obtained by setting e

and θ_e to 0:

$$\begin{aligned}\frac{\partial x_{ref}}{\partial s} &= \cos(\theta_{ref}), \\ \frac{\partial y_{ref}}{\partial s} &= \sin(\theta_{ref}), \\ \frac{\partial \theta_{ref}}{\partial s} &= c(s).\end{aligned} \tag{6.83}$$

Thus, the bounds on the original positions can be directly obtained by computing the Cartesian coordinates of the error e along the normal vector of the curvature C. It follows that the bounds on the left and right of vehicle trajectories can be computed as follows

$$\begin{aligned}x^{L/R} &= e^{L/U}\cos(\theta_{ref} + \pi/2) + x_{ref}, \\ y^{L/R} &= e^{L/U}\sin(\theta_{ref} + \pi/2) + y_{ref},\end{aligned} \tag{6.84}$$

where (x^L, y^L) is the position of the left bounds of trajectories and (x^R, y^R) is the position of the right bounds of trajectories. Let the piecewise inputs be $c(s) = 1/30$ for $s \in [0, 80]$ m and $c(s) = -1/30$ for $s \in [80, 160]$ m. Let the initial condition being uncertain $(e, \theta_e) \in [0.8, 1]$ m $\times\, [\pi/12, \pi/6]$. We firstly compute interval bounds of error states as functions of time. Then, for every time instance t, we compute x_{ref} and y_{ref} for the interval of s corresponding to the time instance t. Thus, we can compute a piecewise function for reference path x_{ref} and y_{ref} with respect to time instance t. Therefore, by (6.84), we can obtain the bounds of positions x and y at every time t. Figure 6.13 samples several time instances t. Then, the bounds are computed for every sampled time t using Method (iv) (purple). In a path tracking problem, we are interested in how far the vehicle deviates from the reference path. Therefore, in order to compute bounds of the positions at a given $\hat{s}$, we need to compute union of the bounds at every time instance t, whose corresponding interval of s contains $\hat{s}$. This adds complexity when applying the algorithm to safety verification.

In order to directly compute bounds on x and y, we propose a coordinate transformation method by multiplying $\frac{dt}{ds} = \frac{1}{\dot{s}}$ at both sides of the equation (6.70). Thus, we have

$$\begin{aligned} \frac{\partial e}{\partial s} &= \tan\theta_e(1 - c(s)e), \\ \frac{\partial \theta_e}{\partial s} &= (1 - c(s)e)\left(\frac{-g_1\theta_e}{v\cos\theta_e} - \frac{g_2 h_2(\theta_e)e}{\cos\theta_e}\right), \end{aligned} \tag{6.85}$$

Let $\mathcal{L}$ be defined as in (6.72) and it follows that

$$\frac{\partial \mathcal{L}}{\partial s} = -\frac{g_1\theta_e^2(1 - c(s)e)}{g_2 v\cos(\theta_e)}. \tag{6.86}$$

In order to compute bounds on (6.85) and (6.86) using redundancy-based DI method with invariant (6.72), we apply Algorithm 14. Algorithm 14 is modified from Algorithm 12. Lines 4–8 in Algorithm 12 are directly applied in Algorithm 14 for refinements based on (6.77). Finally, the enclosures of the right-hand-side functions in (6.85) and (6.86) are computed by lines 6–8.

Algorithm 14 An implementation of $\mathcal{R}$ for Example 3

```
1:  function $\mathcal{R}(t, Z, P)$
2:      $(E \times \Theta_e \times L) \leftarrow Z$ and $V \leftarrow P$
3:      for $i = 1$ to $l$ do
4:          Apply lines 4–8 in Algorithm 12
5:      end for
6:      $\Sigma_1 \leftarrow \tan(\Theta_e)(1 - \bar{c}E)$
7:      $\Sigma_2 \leftarrow (1 - \bar{c}E)\left(\frac{-\sqrt{V^2+0.1}\Theta_e}{V\cos\Theta_e} - \frac{g_2 H_2(\Theta_e)E}{\cos\Theta_e}\right)$
8:      $\Sigma_3 \leftarrow (1 - \bar{c}E)\frac{\sqrt{V^2+0.1}\Theta_e^2}{g_2 V\cos\Theta_e}$
9:  return $\Sigma \leftarrow (\Sigma_1, \Sigma_2, \Sigma_3)$
10: end function
```

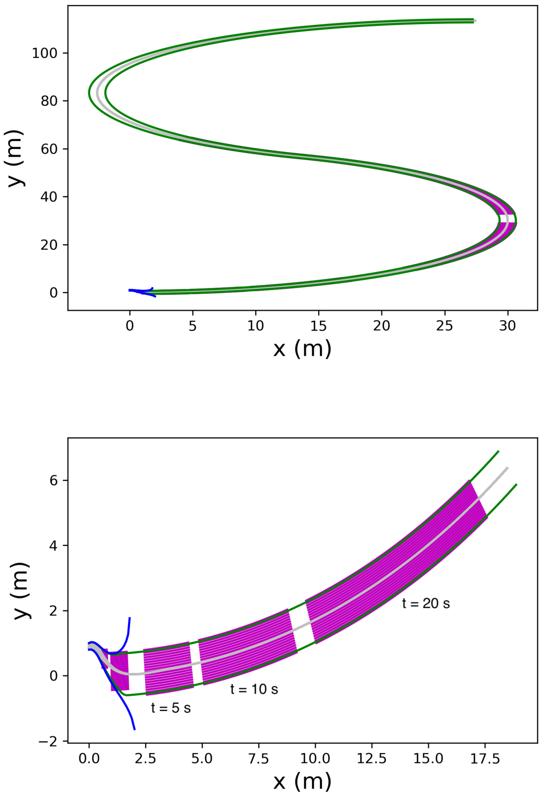

Figure 6.13: Example 3: Bounds on vehicle positions produced by applying standard DI to (6.85) (blue), applying redundancy-based DI to (6.85) and (6.86) with manufactured invariant (6.72) (green), and applying redundancy-based DI to (6.78) and (6.82) with manufactured invariants (6.79) and (6.72) (purple) with 500 sampled trajectories (grey).

The bounds computed on vehicle positions using the DI with invariant method on (6.85) and (6.86), the standard DI method on (6.85), and Method (iv) are compared in Figure 6.13. The standard DI method does not compute effective bounds as expected. Bounds computed directly on (6.85) are continuous and has slightly improvements to Method (iv). Moreover, the redundancy-based DI method with invariants on (6.85) produces very accurate bounds for its original coordinates with the cost of only 0.016 s. This is much more efficient than sampling of 1000 trajectories, which takes more than 0.35 s.

Finally, we prove that Algorithms 12 and 13 satisfy Assumption 6. The proof for

Algorithm 14 is a straightforward extension of the same arguments and is omitted for brevity. Consider the augmented system consisting of (6.70) and (6.73) and let $\mathbf{z} \equiv (s, e, \theta_e, \mathcal{L})$ and $\mathbf{p} = v$ denote generic state and parameter vectors. Similarly, let $Z = S \times E \times \Theta_e \times L$ and $P = V$ denote generic state and parameter interval vectors.

Theorem 20. *Define* $D_{\mathcal{R}} \equiv \{(t, Z, P) \in \mathbb{R} \times \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_p} : E \subset (-|\frac{1}{\bar{c}}|, |\frac{1}{\bar{c}}|), \Theta_e \subset (-\pi/2, \pi/2)\}$. *Algorithm 12 is well defined for every* $(t, Z, P) \in D_{\mathcal{R}}$. *Moreover, Assumption 6 holds with* $\mathcal{R}(t, Z, P)$ *defined by Algorithm 12.*

Proof. Choose any $(t, Z, P) \in D_{\mathcal{R}}$. Algorithm 12 is well-defined for (t, Z, P) if and only if no domain violations occur when evaluating the inclusion functions in lines 4, 5, 6, 7, 8, and 10–13. The only interval operations in lines 4–8 that could possibly lead to a domain violation (i.e., are not defined for every possible argument) are the $\bar{\surd}$ in lines 5 and 7. But by the extended intersections in lines 5 and 7, SQ_E and SQ_{Θ_e} are guaranteed to lie in $\mathbb{IR}^+$, which is the domain of $\bar{\surd}$. Furthermore, the extended intersection in line 6 implies that E is always a subset of $(-|\frac{1}{\bar{c}}|, |\frac{1}{\bar{c}}|)$ and hence the interval $1 - \bar{c}E$ never contains zero. Thus, there is no domain violation in line 10. Similarly, the extended intersection in line 8, Θ_e is always a subset of $(-\pi/2, \pi/2)$, and hence $H_2(\Theta_e)$ is always well defined in line 12. Therefore, $\mathcal{R}$ is well defined on $D_{\mathcal{R}}$.

To verify Condition 1 of Assumption 6, choose any $(t, Z, P) \in D_{\mathcal{R}}$ and let $\Sigma = \mathcal{R}(t, P, Z)$ be the output of Algorithm 12. In order to verify the inclusion in (6.5), we let $\mathbf{f}$ denote the right-hand-side functions of (6.70) and (6.72) and argue that

$$\Sigma \supset \{\mathbf{f}(t, \mathbf{z}, \mathbf{p}) : (\mathbf{z}, \mathbf{p}) \in Z \times P,\ (t, \mathbf{z}, \mathbf{p}) \in G\}, \tag{6.87}$$

where

$$G = \left\{(t, \mathbf{z}, p) \in \mathbb{R}^6 : \mathcal{L} \geq 0, \mathcal{L} = \frac{1}{2}(e^2 + (1/g_2)\theta_e^2)\right\}. \tag{6.88}$$

Choose any $(\mathbf{z}, \mathbf{p}) \in Z \times P$ satisfying $(t, \mathbf{z}, \mathbf{p}) \in G$ and let $(s, e, \theta_e, \mathcal{L}) = \mathbf{z}$. Since $\mathbf{z} \in Z$, the following inclusion all hold immediately after line 2: $s \in S$, $e \in E$, $\theta_e \in \Theta_e$, and $\mathcal{L} \in L$. If these inclusions remain valid when line 10 is reached, then (6.87) must hold because lines 10–13 are direct interval evaluations of the right-hand side functions of (6.70) and (6.73) (i.e., $\mathbf{f}$). Thus, it suffices to show that these inclusions are maintained through lines 4–8. By (6.77), for any $\mathbf{z}, \mathbf{p}$ satisfying $(t, \mathbf{z}, \mathbf{p}) \in G, (\mathbf{z}, \mathbf{p}) \in Z \times P$, we have $\mathbf{z}$ satisfy that $\mathcal{L} \in \frac{1}{2}(E^2 + (1/g_2)\Theta_e^2)$, $e^2 \in 2L - \frac{1}{g_2}\Theta_e^2$, and $\theta_e^2 \in g_2(2L - E^2)$ in lines 4, 5, and 7. Moreover, by the Theorem 9, we have $(e, \theta_e, \mathcal{L}) \in \sqrt[-]{SQ_E} \times \sqrt[-]{SQ_{\Theta_e}} \times L$. By the definition of the extended intersection, $(e, \theta_e, \mathcal{L}) \in E \times \Theta_e \times L$ after lines 4, 6, 8. Therefore, (6.87) holds for the entire algorithm.

Now, we verify Condition 2 of Assumption 6. Choose any $(\hat{t}, \hat{Z}, \hat{P}) \in D_{\mathcal{R}}$. By definition, $\hat{E} \subset (-|\frac{1}{\bar{c}}|, |\frac{1}{\bar{c}}|)$ and $\hat{\Theta}_e \subset (-\pi/2, \pi/2)$. By openness of $(-|\frac{1}{\bar{c}}|, |\frac{1}{\bar{c}}|)$, $(-\pi/2, \pi/2)$, and the definition of the Hausdorff metric, there must exist $\epsilon > 0$ such that $E \subset (-|\frac{1}{\bar{c}}|, |\frac{1}{\bar{c}}|)$ and $\Theta_e \subset (-\pi/2, \pi/2)$ for all $E \times \Theta_e \in \mathbb{IR}^2$ satisfying $d_H(\Theta_e, \hat{\Theta}_e) < \epsilon$ and $d_H(E, \hat{E}) < \epsilon$. By the definition of $D_{\mathcal{R}}$, this implies that $(t, Z, \hat{P}) \in D_{\mathcal{R}}$ for all $t \in B_\epsilon(\hat{t})$ and $Z \in B_\epsilon(\hat{Z})$.

Finally, we verify Condition 3 of Assumption 6 by arguing that every line of the algorithm defines its output as a locally Lipschitz continuous function of all variables on which it depends. It follows that $\mathcal{R}$ is a finite composition of locally Lipschitz functions and is therefore locally Lipschitz.

By Theorem 2.1.1 in [101], the interval operations $+$, $-$, $\times$, x^2, cos, sin, division by a nonzero constant, and square root of a positive constant are all locally Lipschitz continuous on their domains. Moreover, by Theorems 10 and 16, $\sqrt[-]{\ }$ and H_2 are locally Lipschitz continuous on their domains as well. Finally, the extended intersection $\bar{\cap}$ is Lipschitz continuous by Theorem 17. Combining these facts, we conclude that each of the lines 4–8 and 10–13 defines its output by a composition of locally Lipschitz functions, and is therefore locally Lipschitz with respect to all arguments, as desired.

Therefore, the entire algorithm is Lipschitz continuous with respect to (t, Z, P) on all of $\mathcal{D}_{\mathcal{R}}$, which is a stronger condition than Condition 3 of Assumption 6.

$\square$

Similarly, consider the augmented system consisting of (6.78) and let $\mathbf{p} = v$ and $\mathbf{z} \equiv (s, e, \theta_e, \mathcal{L}, \phi_1, \phi_2)$ denote generic state and parameter vectors. Similarly, let $Z = S \times E \times \Theta_e \times L \times \Phi_1 \times \Phi_2$ and $P = V$ denote generic state and parameter interval vectors.

Theorem 21. *Define* $D_{\mathcal{R}} \equiv \{(t, Z, P) \in \mathbb{R} \times \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_p} : E \subset (-|\frac{1}{\bar{c}}|, |\frac{1}{\bar{c}}|), \Theta_e \subset (-\pi/2, \pi/2), V \subset (0, +\infty)\}$*. Algorithm 13 is well defined for every* $(t, Z, P) \in D_{\mathcal{R}}$*. Moreover, Assumption 6 holds with* $\mathcal{R}(t, Z, P)$ *defined by Algorithm 13.*

Proof. Choose any $(t, Z, P) \in D_{\mathcal{R}}$. Algorithm 13 is well-defined for (t, Z, P) if and only if no domain violations occur when evaluating the inclusion functions in lines 4–18 and lines 20–25. Similar as proved in Theorem 20, we have Algorithm 13 well defined for $(t, Z, P) \in \mathbb{R} \times \mathbb{IR}^{n_x} \times \mathbb{IR}^{n_p}$ until line 5. Now, we prove that lines 5–18 have no domain violations. To show this is true for the given definition of $D_{\mathcal{R}}$, choose any $(t, Z, P) \in D_{\mathcal{R}}$ and note that this implies $E \subset (-|\frac{1}{\bar{c}}|, |\frac{1}{\bar{c}}|), \Theta_e \subset (-\pi/2, \pi/2), V \subset (0, +\infty)$. Lines 5, 6, 7 9, 11, and 18 are always evaluated in their appropriate domains. By the extended intersection operators in lines 7, 9, and 11, functions in lines 8, 10, and 12 are always evaluated in $\mathbb{IR}^+$, which is the appropriate domain of the extended square root function. Since V does not contain 0, the extended intersection in line 8 guarantees that the updated V does not contain 0 after line 8. Therefore, lines 13 and 15 do not have domain violations. Moreover, the $\bar{\cap}$ in lines 14, 16, and 18 guarantee that $\Theta_e \subset (-\pi/2, \pi/2)$. Thus, by the extended intersections in lines 13 and 15, $COS_{\Theta_e}, SIN_{\Theta_e} \subset (-1, 1)$. Therefore, functions $\bar{\arccos}$ and $\bar{\arcsin}$ in lines 14 and 16 have no domain violations. Finally, since $\Theta_e \subset (-\pi/2, \pi/2)$ and V does not contain 0, $V \cos(\Theta_e)$ does not contain 0. Thus, by the extended intersection in line

5, Φ_1 does not contain 0 after line 5. The extended intersection operator in line 10 guarantees that Φ_1 does not contain 0. Therefore, there is no domain violation in line 17. Following the proof in Theorem 20, lines 20 – 25 do not have domain violations.

To verify Condition 1 of Assumption 6, choose any $(t, Z, P) \in D_{\mathcal{R}}$ and let $\Sigma = \mathcal{R}(t, P, Z)$ be the output of Algorithm 13. In order to verify the inclusion in (6.5), we let $\mathbf{f}$ denote the right-hand-side of (6.78) and argue that

$$\Sigma \supset \{\mathbf{f}(t, \mathbf{z}, \mathbf{p}) : (\mathbf{z}, \mathbf{p}) \in Z \times P,\ (t, \mathbf{z}, \mathbf{p}) \in G\}, \tag{6.89}$$

where

$$G = \left\{(t, \mathbf{z}, \mathbf{p}) \in \mathbb{R}^8 : \text{(6.79) and (6.72) hold}\right\}. \tag{6.90}$$

Choose any $(\mathbf{z}, \mathbf{p}) \in Z \times P$ satisfying $(t, \mathbf{z}, \mathbf{p}) \in G$ and let $(s, e, \theta_e, \mathcal{L}, \phi_1, \phi_2) = \mathbf{z}$. Since $\mathbf{z} \in Z$, the following inclusion all hold immediately after line 2: $s \in S$, $e \in E$, $\theta_e \in \Theta_e$, $\mathcal{L} \in L$, $\phi_1 \in \Phi_1$, and $\phi_2 \in \Phi_2$. If these inclusions remain valid when line 20 is reached, then (6.89) must hold because lines 20–25 are direct interval evaluations of the right-hand side functions of (6.78) (i.e., $\mathbf{f}$). Thus, it suffices to show that these inclusions are maintained through lines 4–18. By the proof of Theorem 20, this inclusion is true before line 5. By (6.81), for any $\mathbf{z}, \mathbf{p}$ satisfying $(t, \mathbf{z}, \mathbf{p}) \in G, (\mathbf{z}, \mathbf{p}) \in Z \times P$, we have $(s, e, \theta_e, \mathcal{L}, \phi_1, \phi_2) = \mathbf{z}$ satisfying $\phi_1 \in \cos(\Theta_e)V, \phi_2 \in \sin(\Theta_e)V, v^2 \in \Phi_1^2 + \Phi_2^2, \phi_1^2 \in V^2 - \Phi_2^2, \Phi_2^2 \in V^2 - \Phi_1^2, \cos(\theta_e) \in \Phi_1/V, \sin(\theta_e) \in \Phi_2/V, \tan(\theta_e) \in \phi_2/\Phi_1$ by (6.81) in lines 5, 6, 7, 9, 11, 13, 15, and 17. Moreover, by Theorem 9 we have $(\mathcal{L}, \phi_1, \phi_2) \in V \times \Phi_1 \times \Phi_2$ at lines 8, 10, 12. Since $\phi_1 \in \cos(\Theta_e)V$ and $\phi_1 \in \Phi_1$, the extended intersection gives that $\phi_1 \in \cos(\Theta_e)V \bar{\cap} \Phi_1$ at lines 5. Similarly, we have $(\mathcal{L}, \phi_1, \phi_2) \in L \times \Phi_1 \times \Phi_2$ after lines 6, 8, 10, and 12. Therefore, (6.87) holds after line 12. By Theorems 14 and 12, if $\cos(\theta_e) \in COS_{\theta_e}, \sin(\theta_e) \in SIN_{\theta_e}$ and $\theta_e \subset (-\pi/2, \pi/2)$, then we have $\theta_e \in \bar{\arccos}(COS_{\Theta_e})$ and $\theta_e \in \bar{\arcsin}(SIN_{\Theta_e})$ after lines 14 and 16. Since $\theta_e \subset (-\pi/2, \pi/2)$

and $\tan(\theta_e) \in TAN_{\Theta_e}$, we have $\arctan(\tan(\theta_e)) = \theta_e \in \arctan(TAN_{\Theta_e})$. By the definition of the extended intersection again, (6.89) holds after lines 14, 16, and 18. Lines 20–25 are computed by interval arithmetic. As given in the proof of Theorem 20, it follows that Σ_1, Σ_2, Σ_3, and Σ_4 before line 24 are inclusions of σ_1–σ_4 in (6.89). Since $\sigma_3 \in \Sigma_3$, lines 24 and 25 define inclusion function for $\dot{\phi}_1$ and $\dot{\phi}_2$. Thus, we have $\sigma_5 \in \Sigma_5$ and $\sigma_6 \in \Sigma_6$ in lines 24 and 25. Therefore, (6.89) holds for the entire algorithm.

Now, we verify Condition 2 of Assumption 6. Choose any $(\hat{t}, \hat{Z}, \hat{P}) \in D_{\mathcal{R}}$. By definition, $\hat{E} \subset (-|\frac{1}{\bar{c}}|, |\frac{1}{\bar{c}}|)$, $\hat{\Theta}_e \subset (-\pi/2, \pi/2)$. By openness of $(-|\frac{1}{\bar{c}}|, |\frac{1}{\bar{c}}|)$, $(-\pi/2, \pi/2)$, and the definition of the Hausdorff metric, there must exist $\epsilon > 0$ such that $E \subset (-|\frac{1}{\bar{c}}|, |\frac{1}{\bar{c}}|)$ and $\Theta_e \subset (-\pi/2, \pi/2)$ for all $E \times \Theta_e \in \mathbb{IR}^2$ satisfying $d_H(\Theta_e, \hat{\Theta}_e) < \epsilon$ and $d_H(E, \hat{E}) < \epsilon$. By the definition of $D_{\mathcal{R}}$, this implies that $(t, Z, \hat{P}) \in D_{\mathcal{R}}$ for all $t \in B_\epsilon(\hat{t})$ and $Z \in B_\epsilon(\hat{Z})$.

Finally, we verify Condition 3 of Assumption 6 by arguing that every line of the algorithm defines its output as a locally Lipschitz continuous function of all variables on which it depends. It follows that $\mathcal{R}$ is a finite composition of locally Lipschitz functions and is therefore locally Lipschitz.

By Theorem 2.1.1 in [101], the interval operations $+$, $-$, $\times$, x^2, cos, sin, tan, arctan, division by a nonzero constant, and square root of a positive constant are all locally Lipschitz continuous on their domains. Moreover, by Theorems 10, 14, 12, and 16, $\bar{\sqrt{\ }}$, $\overline{\arccos}$, $\overline{\arcsin}$,and H_2 are locally Lipschitz continuous on their domains as well. Finally, the extended intersection $\bar{\cap}$ is Lipschitz continuous by Theorem 17. Combining these facts, we conclude that each of the lines 4–18 and 20–25 defines its output by a composition of locally Lipschitz functions, and is therefore locally Lipschitz with respect to all arguments, as desired. Therefore, the entire algorithm is Lipschitz continuous with respect to (t, Z, P) on all of $\mathcal{D}_{\mathcal{R}}$, which is a stronger condition than Condition 3 of Assumption 6. $\square$

6.5 Conclusion

This chapter proposes strategies for manufacturing invariants for vehicle models under path and trajectory tracking control laws. This allows effective redundancy-based DI methods to be applied to efficiently compute accurate reachability bounds for these models. The first key result of this chapter is that the choice of system coordinates for computing reachable set enclosures is critical. Directly computing reachability bounds in the coordinates where the controller is derived often causes certain nonlinear terms to cancel, which can significantly reduce the conservatism of the computed bounds. The second key result is that adding redundant model equations in the form of Lyapunov-like functions leads to very effective manufacture invariants for this class of problems, which enables the proposed DI method to compute tight reachability bounds. These strategies were applied to three representative path and trajectory tracking examples. In all cases, we ultimately obtained reachability bounds that are greatly improved relative to the standard DI method, and appear both accurate and efficient enough to support many online safety verification tasks. Moreover, custom refinement algorithms based on algebraic rearrangements of these invariants were also proposed, which produce much tighter bounds than using the existing refinement algorithm in [103]. The produced reachable set enclosures can be potentially used for safety verification during vehicle motion planning.

Algorithm 10 An implementation of $\mathcal{R}$

```
**function** $\mathcal{R}(t, Z, P)$
    $(E_t, E_n, E_\theta, K_\delta, V, L, L_C, E_\delta) \leftarrow Z$
    **for** $i = 1$ to $l$ **do**
        $L_C = L_C \bar{\cap} (L + 0.5E_\delta^2)$
        $L = L \bar{\cap} (L_C - 0.5E_\delta^2)$
        $L = L \bar{\cap} [k_1 E_t^2 + k_1 E_n^2 + E_\theta^2 + (V - v_{ref})^2]/2$
        $SQ_{E_\delta} = E_\delta^2 \bar{\cap} 2(L_C - L)$
        $E_\delta = E_\delta \bar{\cap} \sqrt[-]{SQ_{E_\delta}}$
        $SQ_{E_\theta} = E_\theta^2 \bar{\cap} 2(L - k_1 E_t^2 - k_1 E_n^2 - (V - v_{ref})^2)$
        $E_\theta = E_\theta \bar{\cap} \sqrt[-]{SQ_{E_\theta}}$
        $SQ_{E_t} = E_t^2 \bar{\cap} \frac{2}{k_1}(L - E_\theta^2 - k_1 E_n^2 - (V - v_{ref})^2)$
        $E_t = E_t \bar{\cap} \sqrt[-]{SQ_{E_t}}$
        $SQ_{E_n} = E_n^2 \bar{\cap} \frac{2}{k_1}(L - E_\theta^2 - k_1 E_t^2 - (V - V_{ref})^2)$
        $E_n = E_n \bar{\cap} \sqrt[-]{SQ_{E_n}}$
        $SQ_{E_v} = (V - v_{ref})^2 \bar{\cap} 2(L - k_1 E_t^2 - k_1 E_n^2 - E_\theta^2)$
        $E_v = V \bar{\cap} (\sqrt[-]{SQ_{E_v}} + v_{ref})$
        $\Xi = \kappa_{ref} - k_1[E_t H_1(E_\theta) + E_n H_2(E_\theta)] - k_2 E_\theta$
        $E_\delta = E_\delta \bar{\cap} (K_\delta - \Xi)$
        $K_\delta = K_\delta \bar{\cap} (E_\delta + \Xi)$
    **end for**
    $\Sigma_1 \leftarrow V \cos E_\theta - v_{ref}[1 - \kappa_{ref} E_n]$
    $\Sigma_2 \leftarrow V \sin E_\theta - v_{ref} \kappa_{ref} E_t$
    $\Sigma_3 \leftarrow V K_\delta - v_{ref} \kappa_{ref}$
    $\dot{\Xi} \leftarrow \dot{\kappa}_{ref} - k_1[\Sigma_1 H_1(E_\theta) - E_t \Sigma_3 (H_2(E_\theta) + H_3(E_\theta)) + \Sigma_3 H_2(E_\theta) - E_n \Sigma_3 H_4(E_\theta)] - k_2 \Sigma_3$

    $\Sigma_4 \leftarrow -E_\theta V + \dot{\Xi} - k_4(K_\delta - \Xi)$
    $\Sigma_5 \leftarrow \dot{v}_{ref} - k_1 E_t - k_3(V - v_{ref}) + k_2 E_\theta^2 - E_\theta \kappa_{ref}$
    $\Sigma_6 \leftarrow -v_{ref} k_2 E_\theta^2 - k_3(V - v_{ref})^2$
    $\Sigma_7 \leftarrow -v_{ref} k_2 E_\theta^2 - k_3(V - v_{ref})^2 - k_4(K_\delta - \Xi)^2$
    $\Sigma_8 \leftarrow -E_\phi V - k_4 E_\delta$
**return** $\Sigma \leftarrow (\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \Sigma_7, \Sigma_8)$
**end function**
```

CHAPTER 7

CONCLUSION

This thesis has proposed a set-based fault detection algorithm based on a new fast and accurate state estimation method. Specifically, this work firstly developed an effective reachability analysis method for nonlinear discrete-time systems with uncertainties. This is a significant step because enclosing reachable sets is a critical step in set-based state estimation, which is in turn used in a variety of robust control and fault detection algorithms. This reachability algorithm was then extended to set-based state estimation. Finally, a set-based fault detection algorithm was proposed based on the set-based state estimation method. This fault detection method eliminates false alarms and can effectively mitigate the safety risks associated with abnormal operations, as well as the associated economic losses caused by off-spec production, maintenance, and downtime. Moreover, this thesis addressed two more problems as additional contributions. First, existing zonotope order reduction methods were reviewed and compared, providing valuable guidance for designing set-based control algorithms using zonotopes including reachability analysis, state estimation, robust control, and fault detection. Second, a safety verification method was developed for automated vehicles under path or trajectory tracking control using rigorous continuous-time reachable set bounding method.

7.1 Summary of Contributions

Chapter 2 developed a new class of methods for discrete-time reachability analysis motivated by continuous-time methods based on differential inequalities (DI), which is a main theoretical contribution of this work. Specifically, Chapter 2 proves that DI methods can be used to compute reachable sets for discrete-time systems obtained by forward Euler discretization. Focusing on this special case, our main results show that, for any given system, there exists a bound on the discretization step size below which a discrete-time analogue of the basic DI method provides bounds on the reachable sets of the discretized system. This bound depends on Lipschitz constants for the dynamics and can be easily computed in advance. Moreover, this step size is no more restrictive than the step size required to preserve basic physical properties of the solution (i.e. non-negativity). Moreover, Chapter 2 proves that the discrete-time reachability analysis algorithm can be extended to general discrete-time systems as well under sufficient monotonicity conditions. Next, we show that advanced DI methods using manufactured invariants are also valid in discrete time under a tighter step size restriction. Moreover, three refinement operators are proposed for exploiting linear and nonlinear invariants in a way that effectively balances accuracy with the need to achieve a small Lipschitz constant. Compared to the algorithm proposed in [26, 96], the new algorithms result in much smaller Lipschitz constants, which in turn leads to a reasonable step size upper bound for forward Euler discrete-time systems. Additionally, the theoretical development is generalized to consider dynamic systems subject to externally imposed state constraints, where one is only interested in bounding the feasible trajectories, which is useful in optimal control applications. Finally, we compare the proposed methods to the standard discrete-time interval method and two popular methods using zonotopes [22, 23]. The numerical results show that the proposed DI methods in this chapter offer significant advantages in terms of both speed and accuracy, especially for highly nonlinear and uncertain systems. Specifically, Examples 1–4 show that the standard DI method can provide significant gains in accuracy at lower cost when compared with existing bounding approaches based zonotopes, while advanced DI methods using refinements based on redundant model equations provide much more accurate bounds at similar cost. However, Examples 5–6 show that zonotopic approaches are still more effective for some problems. Our results suggest that zonotopic methods are more effective when the interval Jacobian or Hessian matrices used for bounding the linearization errors in these methods have few uncertain elements. In contrast, the DI approaches appear to be more effective for highly nonlinear systems with large uncertainties, particularly when nonlinear or uncertain terms can be canceled through the introduction of appropriate new variables and manufactured invariants.

Chapter 3 develops a new set-based state estimation algorithm by adapting the DI-based reachability method in Chapter 2 to provide accurate prediction sets using only fast interval computations and adding an efficient and accurate correction algorithm. The prediction step of our algorithm is not quite a direct application of the method in Chapter 2. The main contribution of this chapter is using output measurements to modify the prediction step in a simple but nontrivial way, leading to significantly tighter prediction bounds. It is proved in this chapter that the new algorithm exploiting the measurement within DI produces valid enclosures for the predictions with the time step restriction stated in Chapter 3. The numerical results clearly verify the accuracy and efficiency of the proposed prediction algorithm. Moreover, a correction step is combined with the prediction method for guaranteed state estimation. The propose algorithm significantly improves the accuracy of the estimated state sets and is suitable for online applications. Finally, the numerical results show that this method produces state estimates with significantly higher accuracy and efficiency than state-of-the-art zonotopic methods for a challenging nonlinear chemical reactor model.

The major contribution of Chapter 4 is a rapid and accurate guaranteed set-based fault detection method based on the set-based state estimation algorithm in Chapter 3. The proposed algorithm is compared with one data-based method using principal component analysis (PCA), one model-based method using the extended Kalman filter (EKF), and four state-of-the-art set-based algorithms in four numerical case studies. For each case study, these fault detection algorithms are firstly tested in a nominal fault-free scenario. The results show that the FD methods using PCA and EKF generate a small number of false alarms, especially for highly nonlinear and uncertain systems, while the set-based methods have no false alarms as expected. The methods were also tested in fault-free scenarios with large disturbances. In these scenarios, the FD methods using PCA and EKF both generated many false alarms, rendering them ineffective for fault detection. In contrast, the set-based methods again gave no false alarms. Finally, the methods were compared in multiple different faulty scenarios. The proposed FD method using DI detects faults significantly faster than the other set-based methods and is competitive with the detection speed of PCA and EKF for many faults. For systems with large uncertainties, the zonotopic set-based methods failed to detect most faults due to the conservative bounds computed for the prediction step.

Chapter 5 reviews and compares four existing zonotope order reduction methods. This work has two main contributions. First, the order reduction method by [4] is presented in detail and its validity is established. Second, a comprehensive comparison of the existing four methods is presented considering both computational cost and overestimation error for a large test set. The effects of problem dimension, initial zonotope order, and reduced zonotope order are also investigated. The results provide valuable guidance for designing set-based estimation and control algorithms that more effectively balance accuracy with computational cost.

The major contributions of Chapter 6 are strategies for manufacturing invariants for vehicle models under path and trajectory tracking control laws. This allows effective redundancy-based DI methods to be applied to efficiently compute accurate reachability bounds for these models. The first key result of this chapter is that the choice of system coordinates for computing reachable set enclosures is critical. Directly computing reachability bounds in the coordinates where the controller is derived often causes certain nonlinear terms to cancel, which can significantly reduce the conservatism of the computed bounds. The second key result is that adding redundant model equations in the form of Lyapunov-like functions leads to very effective manufacture invariants for this class of problems, which enables the proposed DI method to compute tight reachability bounds. These strategies were applied to three representative path and trajectory tracking examples. In all cases, we ultimately obtained reachability bounds that are greatly improved relative to the standard DI method, and appear both accurate and efficient enough to support many online safety verification tasks. Moreover, custom refinement algorithms based on algebraic rearrangements of these invariants were also proposed, which produce much tighter bounds than using the existing refinement algorithm in [103]. The produced reachable set enclosures can be potentially used for safety verification during vehicle motion planning.

7.2 Future Work

Chapter 2 developed differential inequalities methods for nonlinear discrete-time systems. Although these methods show great advantages in terms of both speed and accuracy, they only produce valid bounds for systems under sufficient monotonicity conditions, or for forward Euler discretized systems with sufficiently small step sizes. These limitations prohibit the proposed methods from being applied to general discrete-time systems, and make their application more cumbersome even for systems where they do apply. Future work should focus on modifying the proposed methods to remove these restrictions while retaining the efficiency and accuracy of these methods. Moreover, while these methods produced the tightest bounds among all compared methods when applied to highly nonlinear systems, existing zonotopic methods still produced tighter bounds for some specific examples, as shown by Examples 5 and 6 in Chapter 2. Therefore, combining the proposed DI methods with zonotopes could potentially lead to significant further improvements.

The major limitation of the state estimation algorithm in Chapter 3 is the lack of a refinement operator for nonlinear invariants, which should be considered in future work. Moreover, the proposed algorithm only applies to discrete-time systems obtained by forward Euler discretization. Future work to extend the estimation algorithm to general discrete-time systems should be considered.

This dissertation mainly focuses on passive set-based fault detection. The proposed algorithm should be extended to fault diagnosis rather than just detection. This can be done by applying the state estimation algorithm in Chapter 3 to faulty models in order to test the consistency of measured outputs with each potential fault. Moreover, although these methods are guaranteed to eliminate false alarms, they are not guaranteed to detect faults when they occur because sets of normal conditions and faulty conditions can be intersected. Therefore, active fault detection methods should be developed to guarantee that no faults are missed. This can be achieved by computing active inputs to separate sets of normal condition and faulty conditions using optimization techniques.

Chapter 6 computes rigorous bounds for vehicle trajectories to verify the safety of automated driving. However, this work does not provide any strategies for changing the controller or motion plan when the verification algorithm indicates that the current plan may cause a collision. Future work should focus on computing safe inputs to avoid potential collisions using backward reachability analysis.

REFERENCES

[1] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. Kavuri, “A review of process fault detection and diagnosis: Part I: Quantitative model-based methods”, *Computers and chemical engineering*, vol. 27, no. 3, pp. 293–311, 2003.

[2] W. Kuhn, “Rigorously computed orbits of dynamical systems without the wrapping effect”, *Computing*, vol. 61, no. 1, pp. 47–67, 1998.

[3] V. T. H. Le, C. Stoica, T. Alamo, E. F. Camacho, and D. Dumur, “Zonotopic guaranteed state estimation for uncertain systems”, *Automatica*, vol. 49, no. 11, pp. 3418–3424, 2013.

[4] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz, “Constrained zonotopes: A new tool for set-based estimation and fault detection”, *Automatica*, vol. 69, pp. 126–136, 2016.

[5] F. D. Brunner, M. Heemels, and F. Allgower, “Robust self-triggered MPC for constrained linear systems: A tube-based approach”, *Automatica*, vol. 72, pp. 73–83, 2016.

[6] L. Zhang, S. Zhuang, and R. D. Braatz, “Switched model predictive control of switched linear systems: Feasibility, stability and robustness”, *Automatica*, vol. 67, pp. 8–21, 2016.

[7] D. M. Raimondo, G. R. Marseglia, R. D. Braatz, and J. K. Scott, “Closed-loop input design for guaranteed fault diagnosis using set-valued observers”, *Automatica*, vol. 74, pp. 107–117, 2016.

[8] S. Tornil-Sin, C. Ocampo-Martinez, V. Puig, and T. Escobet, “Robust fault detection of non-linear systems using set-membership state estimation based on constraint satisfaction”, *Engineering Applications of Artificial Intelligence*, vol. 25, no. 1, pp. 1–10, 2012.

[9] D. Bresolin, L. Geretti, R. Muradore, P. Fiorini, and T. Villa, “Formal verification of robotic surgery tasks by reachability analysis”, *Microprocess. Microsyst.*, vol. 39, no. 8, pp. 836–842, 2015.

[10] M. Althoff and J. M. Dolan, “Online verification of automated road vehicles using reachability analysis”, *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

[11] M. Kishida and R. D. Braatz, “Skewed structured singular value-based approach for the construction of design spaces: Theory and applications”, *IET Control Theory A.*, vol. 8, no. 14, pp. 1321–1327, 2014.

[12] Y. Becis-Aubry, M. Boutayeb, and M. Darouach, “State estimation in the presence of bounded disturbances”, *Automatica*, vol. 44, no. 7, pp. 1867–1873, 2008.

[13] L. Chisci, A. Garulli, and G. Zappa, “Recursive state bounding by parallelotopes”, *Automatica*, vol. 32, no. 7, pp. 1049–1055, 1996.

[14] J. S. Shamma and K. Y. Tu, “Set-valued observers and optimal disturbance rejection”, *IEEE Trans. Autom. Control*, vol. 44, pp. 253–264, 1999.

[15] M. Kieffer, L. Jaulin, and E. Walter, “Guaranteed recursive non-linear state bounding using interval analysis”, *Int. J. Adapt Control Signal Process.*, vol. 16, no. 3, pp. 193–218, 2002.

[16] M. Kieffer and E. Walter, “Interval analysis for guaranteed non-linear parameter and state estimation.”, *Math. Comp. Model Dyn.*, vol. 11, no. 2, pp. 171–181, 2005.

[17] M. A. B. Sassi, R. Testylier, T. Dang, and A. Girard, “Reachability analysis of polynomial systems using linear programming relaxations”, in *Automated Technology for Verification and Analysis*, vol. 7561, 2012, pp. 137–151.

[18] S. Streif, K. K. Kim, P. Rumschinski, M. Kishida, D. E. Shen, R. Findeisen, and R. D. Braatz, “Robustness analysis, prediction, and estimation for uncertain biochemical networks: An overview”, *J. Process Contr.*, vol. 42, pp. 14–34, 2016.

[19] T. Alamo, J. M. Bravo, M. J. Redondo, and E. F. Camacho, “A set-membership state estimation algorithm based on DC programming”, *Automatica*, vol. 44, no. 1, pp. 216–224, 2008.

[20] T. Dreossi, T. Dang, and C. Piazza, “Reachability computation for polynomial dynamical systems”, *Form. Meth. Syst. Des.*, vol. 50, no. 1, pp. 1–38, 2017.

[21] M. Kishida, P. Rumschinski, R. Findeisen, and R. D. Braatz, “Efficient polynomial-time outer bounds on state trajectories for uncertain polynomial systems using skewed structured singular values”, *IEEE Trans. Automat. Contr.*, vol. 59, no. 11, pp. 3063–3068, 2014.

[22] C. Combastel, "A state bounding observer for uncertain non-linear continuous-time systems based on zonotopes", in *Conference on Decision and Control*, 2005, pp. 7228–7234.

[23] T. Alamo, J. M. Bravo, and E. F. Camacho, "Guaranteed state estimation by zonotopes", *Automatica*, vol. 41, no. 6, pp. 1035–1043, 2005.

[24] J. K. Scott and P. I. Barton, "Bounds on the reachable sets of nonlinear control systems", *Automatica*, vol. 49, no. 1, pp. 93–100, 2013.

[25] J. K. Scott and P. I. Barton, "Tight, efficient bounds on the solutions of chemical kinetics models", *Computers and Chemical Engineering*, vol. 34, no. 5, pp. 717–731, 2010.

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

[27] S. M. Harwood and P. I. Barton, "Efficient polyhedral enclosures for the reachable set of nonlinear control systems", *Math. Control Signals Syst*, vol. 28, no. 1, p. 8, 2016.

[28] W. Walter, *Differential and Integral Inequalities.* New York: Springer-Verlag, 1970.

[29] Z. Horvath, "On the positivity step size threshold of Runge-Kutta methods", *Appl. Numer. Math.*, vol. 53, no. 2, pp. 341–356, 2005.

[30] D. Q. Mayne, E. C. Kerrigan, E. J. Van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control", *Int. J. Robust and Nonlinear Control*, vol. 21, no. 11, pp. 1341–1353, 2011.

[31] V. T. H. Le, C. Stoica, D. Dumur, T. Alamo, and E. F. Camacho, "Robust tube-based constrained predictive control via zonotopic set-membership estimation", in *Conference on Decision and Control*, 2011, 4580–4585.

[32] P. Casau, P. Rosa, S. M. Tabatabaeipour, C. Silvestre, and J. Stoustrup, "A set-valued approach to FDI and FTC of wind turbines", *IEEE Transactions on Control Systems Technology*, vol. 23, no. 1, pp. 245–263, 2015.

[33] D. M. Raimondo, G. R. Marseglia, R. D. Braatz, and J. K. Scott, "Fault-tolerant model predictive control with active fault isolation", in *International Conference on Control and Fault Tolerant Systems*, 2013, pp. 444–449.

[34] R. Kianfar, P. Falcone, and J. Fredriksson, “Safety verification of automated driving systems”, *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 4, pp. 73–86, 2013.

[35] C. Durieu, E. Walter, and B. Polyak, “Multi-input multi-output ellipsoidal state bounding”, *J. Optim. Theor. Appl.*, vol. 111, no. 2, pp. 273–303, 2001.

[36] F. Schweppe, “Recursive state estimation: Unknown but bounded errors and system inputs”, *IEEE Trans. Autom. Control*, vol. 13, no. 1, pp. 22–28, 1968.

[37] J. M. Bravo, T. Alamo, and E. F. Camacho, “Bounded error identification of systems with time-varying parameters”, *IEEE Trans. Autom. Control*, vol. 51, pp. 1144–1150, 2006.

[38] I. Nimmo, “Adequately address abnormal operations”, *Chemical Engineering Progress*, vol. 91, no. 9, Sep. 1995.

[39] O. Pettersson, “Execution monitoring in robotics: A survey”, *Robotics and Autonomous Systems*, vol. 53, no. 2, pp. 73–88, 2005.

[40] D. Stavrou, D. G. Eliades, C. G. Panayiotou, and M. M. Polycarpou, “Fault detection for service mobile robots using model-based method”, *Autonomous Robots*, vol. 40, no. 2, pp. 383–394, 2016.

[41] L. H. Chiang, E. L. Russell, and R. D. Braatz, *Fault Detection and Diagnosis in Industrial Systems*. Springer-Verlag London Limited, 2001.

[42] M. M. Seron and J. A. De Doná, “Fault tolerant control using virtual actuators and invariant-set based fault detection and identification”, in *Conference on Decision and Control*, 2009, pp. 7801–7806.

[43] D. Efimov, T. Raïssi, S. Chebotarev, and A. Zolghadri, “Interval state observer for nonlinear time varying systems”, *Automatica*, vol. 49, no. 1, pp. 200–205, 2013.

[44] J. Blesa, V. Puig, and J. Saludes, “Robust fault detection using polytope-based set-membership consistency test”, *IET Control Theory and Applications*, vol. 6, no. 12, pp. 1767–1777, 2012.

[45] V Reppa and A. Tzes, “Fault detection and diagnosis based on parameter set estimation”, *IET control theory and applications*, vol. 5, no. 1, pp. 69–83, 2011.

[46] J. K. Scott, R. Findeisen, R. D. Braatz, and D. M. Raimondo, “Design of active inputs for set-based fault diagnosis”, in *American Control Conference*, 2013, pp. 3561–3566.

[47] A. Ingimundarson, J. M. Bravo, V. Puig, T. Alamo, and P. Guerra, “Robust fault detection using zonotope-based set-membership consistency test”, *International Journal of Adaptive Control and Signal Processing*, vol. 23, no. 4, pp. 311–330, 2009.

[48] S. M. Tabatabaeipour, P. F. Odgaard, T. Bak, and J. Stoustrup, “Fault detection of wind turbines with uncertain parameters: A set-membership approach”, *Energies*, vol. 5, no. 7, pp. 2424–2448, 2012.

[49] J. A. Paulson, D. M. Raimondo, R. Findeisen, R. D. Braatz, and S. Streif, “Guaranteed active fault diagnosis for uncertain nonlinear systems”, in *European Control Conference*, 2014, pp. 926–931.

[50] C. Jauberthie, N. Verdière, and L. Travé-Massuyès, “Fault detection and identification relying on set-membership identifiability”, *Annual Reviews in Control*, vol. 37, no. 1, pp. 129–136, 2013.

[51] R. M. Fernández-Cantí, S. Tornil-Sin, J. Blesa, and V. Puig, “Nonlinear set-membership identification and fault detection using a bayesian framework: Application to the wind turbine benchmark”, in *Conference on Decision and Control*, 2013, pp. 496–501.

[52] T. Raïssi, G. Videau, and A. Zolghadri, “Interval observer design for consistency checks of nonlinear continuous-time systems”, *Automatica*, vol. 46, no. 3, pp. 518 –527, 2010.

[53] C. Combastel, “An extended zonotopic and Gaussian Kalman filter (EZGKF) merging set-membership and stochastic paradigms: Toward non-linear filtering and fault detection”, *Annual Reviews in Control*, vol. 42, pp. 232–243, 2016.

[54] V. Rostampour, R. Ferrari, and T. Keviczky, “A set based probabilistic approach to threshold design for optimal fault detection”, in *American Control Conference*, 2017, pp. 5422–5429.

[55] Y. Wang and V. Puig, “Zonotopic extended kalman filter and fault detection of discrete-time nonlinear systems applied to a quadrotor helicopter”, in *Conference on Control and Fault-Tolerant Systems*, 2016, pp. 367–372.

[56] W. Chai, J. Qiao, and H. Wang, “Robust fault detection using set membership estimation and T-S fuzzy neural network”, in *American Control Conference*, 2013, pp. 893–898.

[57] A. Tulsyan and P. I. Barton, “Reachability-based fault detection method for uncertain chemical flow reactors”, *IFAC-PapersOnLine*, vol. 49, no. 7, pp. 1–6, 2016.

[58] A. Girard, “Reachability of uncertain linear systems using zonotopes”, in *Hybrid Systems: Computation and Control*, 2005, pp. 291–305.

[59] M. Althoff, O. Stursberg, and M. Buss, “Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization”, in *Conference on Decision and Control*, 2008, pp. 4042–4048.

[60] C. Combastel, “A state bounding observer based on zonotopes”, in *Proc. European Control Conference*, 2003.

[61] C. Combastel, “Zonotopes and kalman observers: Gain optimality under distinct uncertainty paradigms and robust convergence”, *Automatica*, vol. 55, pp. 265–273, 2015.

[62] A Girard and C. L. Guernic, “Zonotope/hyperplane intersection for hybrid systems reachability analysis”, in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, vol. 4981, 2008, pp. 215–228.

[63] M. Althoff, O. Stursberg, and M. Buss, “Verification of uncertain embedded systems by computing reachable sets based on zonotopes”, in *Proc. 17th IFAC World Congress*, 2008, pp. 5125–5130.

[64] M. Althoff, O. Stursberg, and M. Buss, “Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes”, *Nonlinear analysis: hybrid systems*, vol. 4, no. 2, pp. 233–249, 2010.

[65] C. Ocampo-Martinez, P. Guerra, V. Puig, and J. Quevedo, “Actuator fault-tolerance evaluation of linear constrained model-predictive control using zonotope-based set computations”, *J. Systems Control and Engineering*, vol. 221, no. 6, pp. 915–926, 2007.

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

[67] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles”, *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[68] J.-M. Yang and J.-H. Kim, “Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots”, *IEEE Transactions on robotics and automation*, vol. 15, no. 3, pp. 578–587, 1999.

[69] A. Broadhurst, S. Baker, and T. Kanade, “Monte carlo road safety reasoning”, in *Intelligent Vehicles Symposium*, 2005, pp. 319–324.

[70] A. Eidehall and L. Petersson, “Statistical threat assessment for general road scenes using monte carlo sampling”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 137–147, 2008.

[71] L. Yang, J. H. Yang, J. Kuchar, and E. Feron, “A real-time Monte Carlo implementation for computing probability of conflict”, in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004, p. 4876.

[72] C. J. Tomlin, I. Mitchell, A. M. Bayen, and M. Oishi, “Computational techniques for the verification of hybrid systems”, *Proceedings of the IEEE*, vol. 91, no. 7, pp. 986–1001, 2003.

[73] I. M. Mitchell and C. J. Tomlin, “Overapproximating reachable sets by hamilton-jacobi projections”, *Journal of Scientific Computing*, vol. 19, no. 1, pp. 323–346, 2003.

[74] Y. Zhou and J. S. Baras, “Reachable set approach to collision avoidance for UAVs”, in *Conference on Decision and Control*, 2015, pp. 5947–5952.

[75] V. Rubies-Royo, D. Fridovich-Keil, S. Herbert, and C. J. Tomlin, “A classification-based approach for approximate reachability”, in *International Conference on Robotics and Automation*, 2019, pp. 7697–7704.

[76] M. Althoff, O. Stursberg, and M. Buss, “Safety assessment of autonomous cars using verification techniques”, in *American Control Conference*, 2007, pp. 4154–4159.

[77] M. Althoff, O. Stursberg, and M. Buss, “Model-based probabilistic collision detection in autonomous driving”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 299–310, 2009.

[78] E. Yel, T. X. Lin, and N. Bezzo, “Reachability-based self-triggered scheduling and replanning of uav operations”, in *NASA/ESA Conference on Adaptive Hardware and Systems*, 2017, pp. 221–228.

[79] K.-Y. Kim, J.-W. Park, and M.-J. Tahk, “UAV collision avoidance using probabilistic method in 3-D”, in *2007 International Conference on Control, Automation and Systems*, 2007, pp. 826–829.

[80] M Tomizuka *et al.*, “Reachability analysis of hybrid lateral control problem for automated heavy-duty vehicles”, in *American Control Conference*, vol. 1, 2001, pp. 1–6.

[81] P. Falcone, M. Ali, and J. Sjoberg, “Predictive threat assessment via reachability analysis and set invariance theory”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1352–1361, 2011.

[82] M. Althoff and J. M. Dolan, “Set-based computation of vehicle behaviors for the online verification of autonomous vehicles”, in *Conference on Intelligent Transportation Systems*, 2011, pp. 1162–1167.

[83] J. A. Cobano, R. Conde, D. Alejo, and A. Ollero, “Path planning based on genetic algorithms and the monte-carlo method to avoid aerial vehicle collisions under uncertainties”, in *International Conference on Robotics and Automation*, 2011, pp. 4429–4434.

[84] M. Prandini and J. Hu, “Application of reachability analysis for stochastic hybrid systems to aircraft conflict prediction”, in *Conference on Decision and Control*, 2008, pp. 4036–4041.

[85] J. Park, D. Kim, Y. Yoon, H. Kim, and K. Yi, “Obstacle avoidance of autonomous vehicles based on model predictive control”, *Journal of Automobile Engineering*, vol. 223, no. 12, pp. 1499–1516, 2009.

[86] Y. Tassa, T. Erez, and E. Todorov, “Synthesis and stabilization of complex behaviors through online trajectory optimization”, in *International Conference on Intelligent Robots and Systems*, 2012, pp. 4906–4913.

[87] 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”, *International Journal of Vehicle Autonomous Systems*, vol. 8, no. 2-4, pp. 190–216, 2010.

[88] M. Althoff, O. Stursberg, and M. Buss, “Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization”, in *Conference on Decision and Control*, 2008, pp. 4042–4048.

[89] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss, “Validated solutions of initial value problems for ordinary differential equations”, *Applied Mathematics and Computation*, vol. 105, no. 1, pp. 21–68, 1999.

[90] M. Berz and K. Makino, “Performance of Taylor model methods for validated integration of ODEs”, in *Applied Parallel Computing. State of the Art in Scientific Computing*, 2006, pp. 65–73.

[91] Y. Lin and M. A. Stadtherr, “Validated solutions of initial value problems for parametric ODEs”, *Appl. Numer. Math.*, vol. 57, no. 10, pp. 1145–1162, 2007.

[92] B. Houska, M. E. Villanueva, and B. Chachuat, “A validated integration algorithm for nonlinear ODEs using Taylor models and ellipsoidal calculus”, in *Conference on Decision and Control*, IEEE, 2013, pp. 484–489.

[93] B. Houska, M. E. Villanueva, and B. Chachuat, “Stable set-valued integration of nonlinear dynamic systems using affine set-parameterizations”, *SIAM Journal on Numerical Analysis*, vol. 53, no. 5, pp. 2307–2328, 2015.

[94] G. W. Harrison, “Dynamic models with uncertain parameters”, in *International Conference on Mathematical Modeling*, vol. 1, 1997, pp. 295–304.

[95] B. Chachuat and M. Villanueva, “Bounding the solutions of parametric ODEs: When Taylor models meet differential inequalities”, in *Computer Aided Chemical Engineering*, vol. 30, 2012, pp. 1307–1311.

[96] K. Shen and J. K. Scott, “Mean value form enclosures for nonlinear reachability analysis”, in *Conference on Decision and Control*, 2018, pp. 7112–7117.

[97] S. M. Tabatabaeipour, “Active fault detection and isolation of discrete-time linear time-varying systems: A set-membership approach”, *International Journal of Systems Science*, vol. 46, no. 11, pp. 1917–1933, 2015.

[98] A. B. Singer and P. I. Barton, “Bounding the solutions of parameter dependent nonlinear ordinary differential equations”, *SIAM J. Sci. Comput.*, vol. 27, pp. 2167–2182, 2006.

[99] M. E. Villanueva, B. Houska, and B. Chachuat, “Unified framework for the propagation of continuous-time enclosures for parametric nonlinear ODEs”, *J. Glob Optim.*, vol. 62, no. 3, pp. 575–613, 2015.

[100] S. M. Harwood, J. K. Scott, and P. I. Barton, “Bounds on reachable sets using ordinary differential equations with linear programs embedded”, *IMA Journal of Mathematical Control and Information*, vol. 33, no. 2, pp. 519–541, 2016.

[101] A. Neumaier, *Interval Methods for Systems of Equations*. Cambridge University Press, 1991.

[102] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*. Philadelphia: Society for Industrial and Applied Mathematics, 2009.

[103] K. Shen and J. K. Scott, “Tight reachability bounds for nonlinear systems using nonlinear and uncertain solution invariants”, *American Control Conference*, 2018.

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

[105] J. K. Scott and P. I. Barton, "Interval bounds on the ssolution of semi-explicit index-one DAEs. part 1: Analysis", *Numer. Math.*, vol. 125, pp. 1–25, 2013.

[106] X. Yang and J. K. Scott, "Efficient reachability bounds for discrete-time nonlinear systems by extending the continuous-time theory of differential inequalities", *American Control Conference*, pp. 6242–6247, 2018.

[107] M. Du and P. Mhaskar, "Isolation and handling of sensor faults in nonlinear systems", *Automatica*, vol. 50, no. 4, pp. 1066–1074, 2014.

[108] D. P. Bertsekas and I. B. Rhodes, "Recursive state estimation for a set-membership description of uncertainty", *IEEE Trans. Autom. Control*, vol. 16, no. 2, pp. 117–128, 1971.

[109] E. Scholte and M. E. Campbell, "A nonlinear set-membership filter for on-line applications", *International Journal of Robust and Nonlinear Control*, vol. 13, no. 15, pp. 1337–1358, 2003.

[110] D. Simon, "Nonlinear kalman filtering", in *Optimal State Estimation.* John Wiley & Sons, Inc., 2006, pp. 393–431.

[111] M. Nørgaard, N. K. Poulsen, and O. Ravn, "New developments in state estimation for nonlinear systems", *Automatica*, vol. 36, no. 11, pp. 1627–1638, 2000.

[112] Z. Fathi and W. F. Ramirez, "Analytical and knowledge-based redundancy for fault diagnosis in process plants", *Process System Engineering*, vol. 39, no. 1, pp. 42–56, 1993.

[113] G. Salinetti and R. J. B. Wets, "On the convergence of sequences of convex sets in finite dimensions", *SIAM Review*, vol. 21, no. 1, pp. 18–33, 1979.

[114] M. Althoff, "An introduction to CORA 2015", in *Proc. EPiC Series in Computer Science*, vol. 34, Cancun, Mexico, 2015, pp. 120–151.

[115] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. Zamarashkin, "How to find a good submatrix", in *Matrix Methods: Theory, Algorithms, Applications*, 2010, pp. 247–256.

[116] A. Y. Mikhalev and I. V. Oseledets, “Iterative representing set selection for nested cross approximation”, *Numerical Linear Algebra with Applications*, vol. 23, no. 2, pp. 230–248, 2016.

[117] K. Shen and J. K. Scott, “Exploiting nonlinear invariants and path constraints to achieve tighter reachable set enclosures using differential inequalities”, *Mathematics of Control, Signals, and Systems*, pp. 1–27, 2020.

[118] B. O. S. Teixeira, J. Chandrasekar, L. A. B. Tôrres, L. A. Aguirre, and D. S. Bernstein, “State estimation for linear and non-linear equality-constrained systems”, *Int. J. Contr.*, vol. 82, no. 5, pp. 918–936, 2009.

[119] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, “Sundials: Suite of nonlinear and differential/algebraic equation solvers”, *ACM Transactions on Mathematical Software (TOMS)*, vol. 31, no. 3, pp. 363–396, 2005.

[120] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, “A stable tracking control method for an autonomous mobile robot”, 1990.

[121] M. Werling, L. Groll, and G. Bretthauer, “Invariant trajectory tracking with a full-size autonomous road vehicle”, *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 758–765, 2010.

[122] C. Samson, “Path following and time-varying feedback stabilization of a wheeled mobile robot”, in *Proceedings of the International Conference on Advanced Robotics and Computer Vision*, vol. 13, 1992, pp. 1–14.

VITA

Xuejiao Yang received her B.S. degree in Chemical Engineering in 2015 from Dalian University of Technology, Dalian, China. She joined Dr. Joseph Scott's lab in 2015 and joined School of Chemical and Biomolecular Engineering at Georgia Institute of Technology in 2019. Her research interests include reachability analysis, fault detection and diagnosis, and safety verification.