

Real-Time Synthesis of Robust Controlled Invariant Sets for Monotone Systems

Yasin Sonmez, Mahmoud Khaled, Majid Zamani, Murat Arcak

Abstract—Safety-critical control of autonomous systems requires formal safety certificates, such as controlled invariant sets, that must be computed online as conditions change. Although standard synthesis algorithms scale poorly with state dimension, monotone dynamical systems with lower-closed safety specifications allow for accelerated computation of controlled invariant sets. In particular, *lazy* fixed-point algorithms exploit monotonicity and track only the antichain basis of the set. However, membership tests and redundancy checks against an evolving basis remain major bottlenecks. We introduce a threshold-function reformulation in which a lower-closed set on a d -dimensional grid is represented by its column heights along a designated axis. This reformulates the greatest-fixed-point iteration as independent one-dimensional binary searches, one per grid column, yielding an embarrassingly parallel iteration with asymptotically lower computational complexity than the lazy fixed-point algorithm. Experiments synthesize invariant sets on 3D grids with 10^9 cells in under 50 ms and 10^{14} cells in under two minutes. We further demonstrate online re-synthesis in a safety-informed model predictive controller example.

I. INTRODUCTION

Controlled invariant sets characterize safe regions in which trajectories can be kept for all future time through appropriate control actions. Symbolic abstractions [1] locate controlled invariant sets through finite-state fixed-point computations, but their cost grows sharply with state dimension.

One approach to overcome the scalability barrier is to exploit monotone dynamics: When the dynamics are monotone with respect to a partial order and the safe set is lower-closed with respect to the same order, the maximal robust controlled invariant set is also lower-closed and is uniquely determined by its *antichain basis* [2]. On an N^d grid, the basis size is $O(N^{d-1})$, one order of magnitude smaller than the underlying grid. The *lazy* fixed-point algorithms of [2], [3], [4] track only the basis and update the antichain boundary between iterations. Two bottlenecks remain despite this reduction:

- 1) **Quadratic per-iteration cost.** Each iteration performs $O(N^{d-1})$ safety checks, and each check scans the basis to test membership in its lower closure, giving $O(N^{2(d-1)})$ work per iteration.
- 2) **Sequential neighbor generation.** After unsafe elements are removed, new candidate neighbors must be added to the basis, each requiring a redundancy check—an inherently sequential step that limits parallelism.

Instead of representing the boundary of the lower-closed set with basis points, which must be iteratively updated and searched, in this paper we store this surface as a threshold function along a designated axis. We then reformulate the greatest-fixed-point (GFP) iteration to operate *solely* on this threshold function, eliminating basis tracking and sequential neighbor generation. The contributions are:

- We encode lower-closed sets by a threshold function with $O(N^{d-1})$ storage and $O(1)$ membership queries. Parallel column updates yield per-iteration work $O(N^{d-1} \log N)$ without basis enumeration or neighbor checks.
- We prove one-step equivalence with the predecessor operator and equivalence with the lazy basis formulation.
- We show experimentally that the threshold iteration yields speedups on the order of thousands over basis-based lazy synthesis. The 10^{14} -cell scaling benchmarks complete in under two minutes, while the separate 10^{10} -cell closed-loop experiment completes in approximately 62–77 ms.

Existing synthesis methods that leverage structural properties include sparse and compositional constructions for monotone systems and directed specifications [5], flexible abstraction-based pipelines [6], lazy exploration of exposed boundaries using antichain representations [4], and ordered reductions for robust controlled invariance [7]. Height encodings of staircase sets and discrete Pareto frontiers are well-established. Our contribution lies not in the height encoding, but in its use as the sole representation in robust invariant-set GFP iteration, together with the column-prefix result, binary-search update, equivalence proof, and parallel implementation.

II. PRELIMINARIES

A. Monotone systems and lower sets

We study continuous-time systems of the form

$$\dot{\xi} = f(\xi, u, \vartheta), \quad (1)$$

where $\xi \in \mathbb{R}^d$ is the continuous state, $u \in \mathbb{R}^m$ is the control input, and ϑ collects exogenous effects such as disturbances and uncertain parameters. We obtain a finite model from (1) using sampled-data symbolic abstraction [1], [8], [9]. Given a sampling period, state grid, and finite input alphabet, we overestimate one-step reachable sets under all admissible exogenous signals and obtain a finite transition system,

$$x^+ = F(x, u, \delta). \quad (2)$$

Here, x belongs to a finite grid X , u belongs to a finite admissible control set $\mathcal{U}$, and δ belongs to a finite adversarial set

Y. Sonmez and M. Arcak are with the University of California, Berkeley, USA ({yasin_sonmez, arcak}@berkeley.edu); M. Zamani is with the University of Colorado Boulder, USA (majid.zamani@colorado.edu); M. Khaled is with the Ludwig-Maximilian University, Germany (m.mahmoud@lmu.de). This work is supported in part by the NSF grant CNS-2111688.

Δ . This set is finite at the abstraction level and conservatively indexes all abstract successor cells induced by every admissible continuous exogenous signal ϑ , uncertain parameter, and discretization over-approximation over one sampling interval. It is not an arbitrary finite sample of a continuous disturbance set. Replacing a continuous disturbance set by finitely many extremal modes is sound only under an additional dominance property, such as disturbance-state monotonicity.

Definition 1 (Componentwise order). *We equip the state grid with the componentwise order: $x \preceq y$ if and only if $x_j \leq y_j$ for all $j \in \{1, \dots, d\}$. When partial orders on $\mathcal{U}$ and Δ are used, they are denoted by $\preceq_{\mathcal{U}}$ and $\preceq_{\Delta}$.*

Assumption 1 (Finite grid abstraction). *The abstract state space is the finite partially ordered set*

$$X = \{1, \dots, N_1\} \times \dots \times \{1, \dots, N_d\},$$

with the componentwise order of Definition 1. We write $N := \max_j N_j$ and $|X| = \prod_j N_j$.

Definition 2 (Lower set and antichain basis). *For $S \subseteq X$, define its downward closure by*

$$\downarrow S := \{q \in X : \exists s \in S \text{ such that } q \preceq s\}.$$

A set $K \subseteq X$ is a lower set if $q \in K$ and $q' \preceq q$ imply $q' \in K$. Its antichain basis is the set of its maximal elements:

$$\text{Bas}(K) := \{q \in K : \nexists q' \in K \setminus \{q\}, q \preceq q'\}.$$

A lower set K is fully determined by its basis: $K = \downarrow \text{Bas}(K)$.

Remark 1 (Antichain bound). *On the grid of Assumption 1, the maximal antichain size is bounded by $O(N^{d-1})$; every lower set therefore admits a basis of size at most $O(N^{d-1})$, one order of magnitude smaller than $|X| = O(N^d)$.*

B. Monotone dynamics and the robust controlled pre-operator

Definition 3 (Restricted robust controlled predecessor). *For $A, B \subseteq X$ and $V \subseteq \mathcal{U}$, define*

$$\text{Pre}(A, V, B) := \{q \in A : \exists u \in V \forall \delta \in \Delta, F(q, u, \delta) \in B\}. \quad (3)$$

For the safety iteration, define

$$\Phi(K) := \text{Pre}(K, \mathcal{U}, K),$$

which extracts from K those states that can be kept in K for one step despite the disturbances.

Definition 4 (Controlled invariant set). *A subset $K \subseteq X_S$ is controlled invariant (with respect to system (2) and safe set X_S) if for every $x \in K$, there exists $u \in \mathcal{U}$ such that $F(x, u, \delta) \in K$ for all $\delta \in \Delta$. Equivalently, $K = \Phi(K)$.*

Assumption 2 (Lower-closed safety). *The safe state set $X_S \subseteq X$ is a lower set under $\preceq$.*

Definition 5 (State-monotonicity (SM)). *The abstract transition map (2) is state-monotone if*

$$x_1 \preceq x_2 \implies F(x_1, u, \delta) \preceq F(x_2, u, \delta)$$

for all $u \in \mathcal{U}$ and all $\delta \in \Delta$.

Remark 2 (Stronger monotonicity variants). *Reference [7] also equips the control set $\mathcal{U}$ and/or the disturbance set Δ with partial orders to define control-state monotone (CSM), disturbance-state monotone (DSM), and control-disturbance-state monotone (CDSM) systems. The correctness results below require only state monotonicity (SM). When the stronger variants hold, they allow the predecessor check to be restricted to minimal controls, maximal adversarial modes, or both.*

Proposition 1 (Lower-closed predecessor). *[4, § 4.1, Lemma 2] Suppose Assumptions 1 and 2 hold and F is state-monotone (Definition 5). Then $\Phi(K)$ is a lower set for every lower set $K \subseteq X$.*

Proof. Let $x \in \Phi(K)$ and $y \preceq x$. Since $x \in K$ and K is lower, $y \in K$. Choose $u \in \mathcal{U}$ such that $F(x, u, \delta) \in K$ for all $\delta \in \Delta$. By state monotonicity,

$$F(y, u, \delta) \preceq F(x, u, \delta) \in K \quad \forall \delta \in \Delta.$$

Lower-closedness of K then gives $F(y, u, \delta) \in K$ for all δ , hence $y \in \Phi(K)$. $\square$

Corollary 1 (Ordered reductions). *[7, § 5, Thm. 1 and Prop. 2] Let $K \subseteq X$ be a lower set and let $x \in K$. If the stronger CSM variant of Remark 2 holds, then it is enough to test minimal controls:*

$$x \in \Phi(K) \iff \exists u \in \text{Min}(\mathcal{U}) \forall \delta \in \Delta : F(x, u, \delta) \in K.$$

If the stronger DSM variant holds, then for every fixed $u \in \mathcal{U}$ it is enough to test maximal adversarial modes:

$$x \in \Phi(K) \iff \exists u \in \mathcal{U} \forall \delta \in \text{Max}(\Delta) : F(x, u, \delta) \in K.$$

If the CDSM variant holds, one may therefore test only minimal controls against maximal adversarial modes. Let C denote the per-state successor-evaluation count for testing $x \in \Phi(K)$. In the general state-monotone case, $C = |\mathcal{U}| |\Delta|$. The ordered reductions [7] give $C_{\text{CSM}} = |\text{Min}(\mathcal{U})| |\Delta|$, $C_{\text{DSM}} = |\mathcal{U}| |\text{Max}(\Delta)|$, and $C_{\text{CDSM}} = |\text{Min}(\mathcal{U})| |\text{Max}(\Delta)|$.

Theorem 1 (Descending predecessor sequence). *Suppose Assumptions 1 and 2 hold and F is state monotone. Define*

$$K_0 := X_S, \quad K_{i+1} := \Phi(K_i), \quad i \in \mathbb{N}.$$

Then each K_i is a lower set, the sequence is descending, and it converges in finitely many steps to a lower set $K_ \subseteq X_S$ which is the maximal robust controlled invariant subset of X_S .*

Proof. Proposition 1 gives lower-closedness of each K_i by induction. Since $\Phi(K) \subseteq K$ for every K , the sequence is descending. Because X is finite, the descending sequence converges after finitely many strict inclusions.

If $K \subseteq X_S$ is a controlled invariant set, then $K \subseteq K_0$. Assume inductively that $K \subseteq K_i$. Since $K \subseteq \Phi(K)$ and the map Φ is monotone with respect to set inclusion,

$$K \subseteq \Phi(K) \subseteq \Phi(K_i) = K_{i+1}.$$

Hence $K \subseteq K_i$ for all i , and therefore $K \subseteq K_*$. Thus K_* is the maximal robust controlled invariant subset of X_S . $\square$

Algorithm 1 Lazy fixed point iteration [2]

Require: Lower set X_S , transition F , inputs $\mathcal{U}$, adversarial set Δ

```

1:  $B \leftarrow \text{Bas}(X_S)$ 
2: repeat
3:    $B_{\text{pass}} \leftarrow B$ 
4:    $B_u \leftarrow \emptyset$ 
5:   for all  $b \in B_{\text{pass}}$  do ▷ safety check
6:     if  $\nexists u \in \mathcal{U}$  s.t.  $\forall \delta \in \Delta : F(b, u, \delta) \in \downarrow B$  then
7:        $B_u \leftarrow B_u \cup \{b\}$ 
8:        $B \leftarrow B \setminus \{b\}$ 
9:       for all  $j \in \{1, \dots, d\}$  do ▷ neighbor gen.
10:         $c \leftarrow b - e_j$ 
11:        if  $c_j \geq 1$  and  $c \notin \downarrow B$  then
12:           $B \leftarrow B \cup \{c\}$ 
13: until  $B_u = \emptyset$ 
14: return  $\downarrow B$ 

```

C. Lazy fixed-point iteration

Proposition 1 together with Remark 1 means that K_* and every iterate K_i can be represented by its $O(N^{d-1})$ -size antichain basis rather than by the full $O(N^d)$ grid. The lazy algorithm of [2], summarized in Algorithm 1, exploits this by storing only the current basis B and updating it.

Let R_{lazy} denote the number of complete outer basis passes until convergence and $|B| = O(N^{d-1})$ denote the basis-size bound. If the membership query “ $x \in \downarrow B$ ” scans B , the safety-check phase costs $O(R_{\text{lazy}}CN^{2(d-1)})$, up to dimension factors. Neighbor generation performs dominance rather than successor queries and contributes $O(R_{\text{lazy}}N^{2(d-1)})$ with the same scan-based representation. Since an unsafe basis element is removed and neighbors are inserted immediately, subsequent safety and neighbor checks use an evolving B and are sequential across basis updates. For a removed element b , the explicit update in Algorithm 1 implements $\text{Bas}(\downarrow B \setminus \{b\})$ from [2].

III. THRESHOLD-FUNCTION ITERATION

Section III-A introduces the threshold-function representation of lower sets. Section III-B reformulates one GFP step as a parallel collection of binary searches. Section III-C shows equivalence with Algorithm 1 and analyzes complexity.

A. The threshold-function representation

The main idea is to project a d -dimensional lower set $K \subseteq X$ into a $(d-1)$ -dimensional table using a designated axis.

Definition 6 (Designated axis, projection, threshold function). Fix a designated axis $d^* \in \{1, \dots, d\}$ and set

$$\mathcal{K} := \prod_{j \neq d^*} \{1, \dots, N_j\}.$$

For $x \in X$ let $\pi(x) \in \mathcal{K}$ be its projection onto the remaining $d-1$ axes, and write x_{d^*} for its coordinate along d^* . For a lower set $K \subseteq X$, the threshold function is

$$\tau_K(k) := \max\{x_{d^*} : x \in K, \pi(x) = k\}, \quad k \in \mathcal{K}, \quad (4)$$

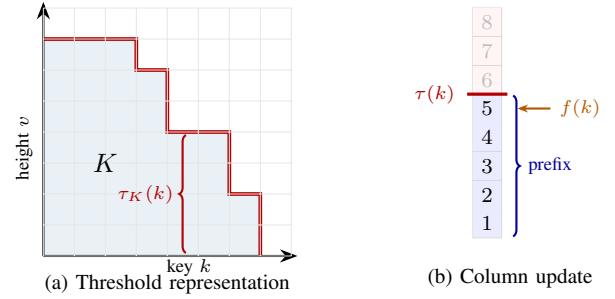


Fig. 1. Threshold representation and update. (a) A lower set K is stored by the column height $\tau_K(k)$, so membership is the $O(1)$ test $(k, v) \in K \iff v \leq \tau_K(k)$. (b) For each column, $\Phi(K)$ is a prefix whose height $f(k)$ is found by binary search; all columns are updated independently.

with the convention $\max \emptyset = 0$.

Each $k \in \mathcal{K}$ indexes a one-dimensional column of cells along the designated axis; lower-closedness forces the safe cells in that column to form a contiguous prefix $\{1, \dots, \tau_K(k)\}$. The function τ_K therefore stores the entire set K in a single integer per column as depicted in Figure 1.

Proposition 2 (Threshold characterization). Let $K \subseteq X$ be a lower set. Then

$$x \in K \iff x_{d^*} \leq \tau_K(\pi(x)).$$

Moreover, τ_K is antitone: $k \preceq k'$ implies $\tau_K(k) \geq \tau_K(k')$.

Proof. If $x \in K$, the value x_{d^*} is one of the values maximized in (4) at $k = \pi(x)$, so $\tau_K(\pi(x)) \geq x_{d^*}$. Conversely, if $x_{d^*} \leq \tau_K(\pi(x))$, pick $y \in K$ with $\pi(y) = \pi(x)$ and $y_{d^*} = \tau_K(\pi(x))$; then $x \preceq y$ and lower-closedness gives $x \in K$. Antitonicity follows by the same lifting: for $k \preceq k'$, any $x \in K$ with $\pi(x) = k'$ and $x_{d^*} = v$ dominates y with $\pi(y) = k$, $y_{d^*} = v$, hence $y \in K$ and $\tau_K(k) \geq v$. $\square$

Remark 3 (Memory and query complexity). τ_K has $|\mathcal{K}| = \prod_{j \neq d^*} N_j = O(N^{d-1})$ entries. Storage is $O(N^{d-1})$ and every inclusion query is $O(1)$ (one projection and one comparison), compared to $O(N^{d-1})$ for a scan-based basis check.

B. The threshold iteration

The representation of Section III-A turns one GFP step into a set of independent binary searches along the designated axis. We first state the structural property that enables this (Lemma 1), then derive the per-column update (Corollary 2).

Definition 7 (Grid column). For $k \in \mathcal{K}$, the column at k is $\text{Col}(k) := \{(k, v) : v \in \{1, \dots, N_{d^*}\}\}$.

Lemma 1 (Column-wise predecessor prefix). If $K \subseteq X$ is a lower set and $(k, v) \in \Phi(K)$, then $(k, v') \in \Phi(K)$ for every $v' \in \{1, \dots, v\}$.

Proof. Let $u \in \mathcal{U}$ be a witness for $(k, v) \in \Phi(K)$, so $F((k, v), u, \delta) \in K$ for every $\delta \in \Delta$. Since $(k, v') \preceq (k, v)$, monotonicity yields $F((k, v'), u, \delta) \preceq F((k, v), u, \delta) \in K$,

Algorithm 2 Threshold-function iteration

Require: Lower set X_S , transition F , inputs $\mathcal{U}$, adversarial set Δ , designated axis d^*

$\tau \leftarrow \tau_{X_S}$

repeat

for all $k \in \mathcal{K}$ **do** $\triangleright$ all columns are independent

$f(k) \leftarrow \text{BINSEARCH}(k, \tau, F, \mathcal{U}, \Delta)$ $\triangleright$ using (5)

$\tau \leftarrow f$

until τ unchanged

return $K_\tau = \{x \in X : x_{d^*} \leq \tau(\pi(x))\}$

and lower-closedness of K gives $F((k, v'), u, \delta) \in K$. Hence $(k, v') \in \Phi(K)$. $\square$

Lemma 1 says that, within any column, the set of states from which K can be maintained is a contiguous prefix $\{(k, 1), \dots, (k, f(k))\}$. This height is efficiently computable:

Corollary 2 (Per-column threshold update). *Let $K \subseteq X$ be lower with threshold function $\tau = \tau_K$. For each $k \in \mathcal{K}$, define*

$$f(k) := \max \left\{ v \in \{1, \dots, \tau(k)\} : \exists u \in \mathcal{U} \forall \delta \in \Delta, F((k, v), u, \delta) \in K \right\}, \quad (5)$$

with $\max \emptyset := 0$. Then

$$\Phi(K) \cap \text{Col}(k) = \{(k, 1), \dots, (k, f(k))\}.$$

Moreover, $f(k)$ can be computed by binary search over $\{1, \dots, \tau(k)\}$. Each probe requires one membership query in K per pair (u, δ) , so the cost per column is $O(C \log N)$.

Proof. By Lemma 1, $\Phi(K) \cap \text{Col}(k)$ is a prefix of $\text{Col}(k)$ of length equal to (5) and the predicate in (5) is monotone non-increasing in v , so a standard binary search over $\{1, \dots, \tau(k)\}$ locates $f(k)$ in $O(\log N)$ probes. $\square$

Since the update $\tau(k) \leftarrow f(k)$ is independent across columns, all columns can be processed in parallel as in Figure 1. The full iteration is summarized in Algorithm 2.

C. Correctness, convergence, and complexity

We now prove that Algorithm 2 computes the same invariant set as Algorithm 1.

Theorem 2 (One-step equivalence). *Let $K \subseteq X$ be a lower set with threshold $\tau = \tau_K$, and let f be defined by (5). Then, Assumption 1 and state monotonicity (Definition 5) imply:*

$$\Phi(K) = \{x \in X : x_{d^*} \leq f(\pi(x))\}, \text{ i.e. } f = \tau_{\Phi(K)}.$$

Proof. By Corollary 2, $\Phi(K) \cap \text{Col}(k) = \{(k, 1), \dots, (k, f(k))\}$ for every $k \in \mathcal{K}$; taking the union over k gives $\Phi(K) = \{x : x_{d^*} \leq f(\pi(x))\}$. Combining this with Proposition 2 yields $f = \tau_{\Phi(K)}$. $\square$

Corollary 3 (Correctness and convergence of Algorithm 2). *Let $\tau_0 := \tau_{X_S}$ and, for each $i \in \mathbb{N}$, let τ_{i+1} be obtained from τ_i by the update (5). Define*

$$K_i := \{x \in X : x_{d^*} \leq \tau_i(\pi(x))\}.$$

TABLE I
PER-ITERATION COMPLEXITY COMPARISON.

	Lazy [2]	Threshold (ours)
Predecessor tests	$O(CN^{2(d-1)})$ sequential over evolving B	$O(CN^{d-1} \log N)$ parallel over columns
Basis update	$O(N^{2(d-1)})$ sequential	—

Then $K_0 = X_S$, each K_i is a lower set, and

$$K_{i+1} = \Phi(K_i) \quad \forall i \in \mathbb{N}.$$

The sequence $(K_i)_{i \in \mathbb{N}}$ is descending and converges in finitely many steps to the maximal robust controlled invariant subset K_* of X_S . Equivalently, Algorithm 2 returns K_* .

Proof. The identity $K_0 = X_S$ follows from the definition of τ_{X_S} together with Proposition 2. Since X_S is a lower set and Theorem 2 maps the threshold of a lower set to the threshold of its predecessor, each K_i is a lower set and satisfies $K_{i+1} = \Phi(K_i)$. The conclusion now follows from Theorem 1. $\square$

Theorem 3 (Equivalence with the lazy basis formulation). *Let K^{lazy} denote the set returned by Algorithm 1 and let K^{thr} denote the set returned by Algorithm 2, both started from the same lower set X_S . Then $K^{\text{lazy}} = K^{\text{thr}} = K_*$.*

Proof. By Corollary 3, Algorithm 2 returns the maximal robust controlled invariant subset K_* of X_S . By the completeness result for the lazy algorithm in [2], Algorithm 1 returns the same maximal invariant set. $\square$

The threshold iteration replaces quadratic lazy boundary work with one embarrassingly parallel stage; see Table I.

Remark 4 (Dense storage and designated-axis choice). *Let $M_{d^*} := \prod_{j \neq d^*} N_j$. A threshold buffer stores M_{d^*} values regardless of frontier complexity, whereas a sparse basis stores $|B|$ maximal grid points and may require substantially less memory when $|B| \ll M_{d^*}$. The threshold representation is attractive for dense or irregular frontiers and workloads dominated by membership queries. Memory can be minimized by choosing an axis with maximal N_{d^*} .*

IV. EXPERIMENTS

We evaluate threshold iteration on three $d=3$ benchmarks and a $d=5$ benchmark from the monotone-synthesis literature.

ACC — adaptive cruise control [2]. The 3D state is (h, v_e, v_ℓ) (headway, ego speed, lead speed), with ego torque command u and adversarial mode δ that captures lead-vehicle braking and parameter uncertainty. The ACC dynamics are

$$\dot{h} = v_\ell - v_e, \quad \dot{v}_e = a_e(v_e, u, \delta), \quad \dot{v}_\ell = a_\ell(v_\ell, \delta).$$

ACC-5D uses state $(h, v_e, v_\ell, T_e, T_\ell)$, replaces direct torque arguments by actuator states in a_e, a_ℓ , and adds $\dot{T}_e = (u - T_e)/\tau_e$, $\dot{T}_\ell = (\delta T - T_\ell)/\tau_\ell$. The acceleration model is

$$a(v, T, \theta) = \frac{1}{M} \left(\frac{T}{R_w} - \alpha \right) - \frac{\beta}{M} v - \frac{\gamma}{M} v^2, \quad (6)$$

where $\theta = (M, R_w, \alpha, \beta, \gamma) \in \Theta$. The safe set is $K_{\text{ACC}} = \{(h, v_e, v_\ell) : h \geq d_b(v_e)\}$, where d_b is the worst-case braking-distance map. We use the orthant order

$$x \preceq_{\text{ACC}} y \iff h_x \geq h_y, v_{e,x} \leq v_{e,y}, v_{\ell,x} \geq v_{\ell,y},$$

and, for ACC-5D, additionally $T_{e,x} \leq T_{e,y}$ and $T_{\ell,x} \geq T_{\ell,y}$. K_{ACC} is lower-closed whenever d_b is nondecreasing.

Turn-Ego/Turn-Oncoming. Both unprotected-turn benchmarks [10] use state (s_e, v_e, s_o) : ego longitudinal position and speed, plus oncoming longitudinal position along the opposing lane centerline. There is no exogenous disturbance channel, so we use a singleton adversarial set. The dynamics are

$$\dot{s}_e = v_e, \quad \dot{v}_e = a_e(v_e, u), \quad \dot{s}_o = v_o.$$

Let $Z(s) := 1 + \mathbf{1}_{\{s \geq -10\}} + \mathbf{1}_{\{s \geq 10\}}$ be the shared zone map and write $Z_e := Z(s_e)$, $Z_o := Z(s_o)$. Both benchmarks use the same unprotected-turn geometry for opposite maneuvers: Turn-Oncoming models the ego waiting while the oncoming car passes, Turn-Ego models the ego passing first; the orthant order is flipped between them, thus the ego-priority and oncoming-priority safe sets are

$$K_{\text{ego}} = \{x \in X_{\text{turn}} : \neg(Z_e < Z_o \vee (Z_e = 2 \wedge Z_o = 2))\},$$

$$K_{\text{onc}} = \{x \in X_{\text{turn}} : \neg(Z_o < Z_e \vee (Z_e = 2 \wedge Z_o = 2))\}.$$

They forbid simultaneous conflict-zone occupation while enforcing the chosen priority relation. The orthant orders are

$$x \preceq_E y \iff s_{e,x} \geq s_{e,y}, v_{e,x} \geq v_{e,y}, s_{o,x} \leq s_{o,y},$$

$$x \preceq_O y \iff s_{e,x} \leq s_{e,y}, v_{e,x} \leq v_{e,y}, s_{o,x} \geq s_{o,y}.$$

Minimal controls are maximum torque for $\preceq_E$ and minimum torque for $\preceq_O$. Since Z is monotone, moving downward under $\preceq_E$ advances ego and delays the oncoming phase, while downward movement under $\preceq_O$ has the reverse effect; thus, priority constraints are lower-closed, and monotonicity follows from the dynamics' sign structure [10].

A. Lazy vs. threshold comparison

For fair comparison, we also introduce a stronger Lazy+ τ baseline which preserves Algorithm 1's traversal and immediate-deletion schedule while maintaining the exact threshold table $\tau_{\downarrow B}$. Deleting an uncontrolled maximal element b updates $\tau(\pi(b)) \leftarrow \tau(\pi(b)) - 1$. A neighbor $c = b - e_j$ is maximal when $c \in K_\tau$ and every valid immediate successor $c + e_\ell \notin K_\tau$, which is checked using threshold queries. Thus, Lazy+ τ changes the representation and boundary-maintenance operations while retaining the sequential Lazy schedule; the basis is materialized after each pass. Table II compares scan-based Lazy, Lazy+ τ , and threshold GFP. The two Lazy methods use the same pass, immediate-deletion schedule, and the same precomputed successor cache. The time required to construct this cache is reported as t_{pre} . The implementation and experiment configurations are publicly available.¹

The threshold method uses inline dynamics: a full successor table would cost $O(C|X|)$ memory and becomes the bottleneck at larger grids, while threshold membership needs only

TABLE II
LAZY VS. THRESHOLD COMPARISON.
 t_{pre} IS THE SUCCESSOR-TABLE GENERATION TIME USED IN LAZY METHODS AND IS EXCLUDED FROM THE REPORTED SPEEDUPS, WHICH EQUAL $t_{\text{lazy}+\tau}/t_{\text{thr}}$; R = ITERATIONS; TIMEOUT = 20 MINUTES.

Case	X	t_{pre}	Lazy			Threshold		
			R_{lazy}	t_{lazy}	$t_{\text{lazy}+\tau}$	R_{thr}	t_{thr}	Speedup
ACC	10^6	.569 ms	182	63.1 s	106 ms	26	3.18 ms	$33.4 \times$
ACC	10^9	439 ms	1595	Timeout	126 s	24	16.9 ms	$7,458 \times$
ACC	10^{10}			Timeout		24	318 ms	-
ACC	10^{12}			Timeout		24	750 ms	-
ACC	10^{14}			Timeout		24	7.94 s	-
ACC-5D	10^6	.624 ms	318	299 s	457 ms	25	3.34 ms	$137 \times$
ACC-5D	10^9	470 ms	835	Timeout	410 s	31	143 ms	$2,870 \times$
ACC-5D	10^{11}			Timeout		28	4.94 s	-
T-Ego	10^6	1.35 ms	278	329 s	269 ms	22	10.8 ms	$24.9 \times$
T-Ego	10^9	1.18 s	2691	Timeout	341 s	26	41.6 ms	$8,188 \times$
T-Ego	10^{10}			Timeout		26	274 ms	-
T-Ego	10^{12}			Timeout		26	3.18 s	-
T-Ego	10^{14}			Timeout		26	70.0 s	-
T-Onc	10^6	.563 ms	273	118 s	173 ms	35	4.22 ms	$41.0 \times$
T-Onc	10^9	458 ms	2462	Timeout	185 s	35	26.4 ms	$6,993 \times$
T-Onc	10^{10}			Timeout		35	123 ms	-
T-Onc	10^{12}			Timeout		34	960 ms	-
T-Onc	10^{14}			Timeout		35	19.2 s	-

the $O(N^{d-1})$ table and eliminates antichain maintenance. At $|X| = 10^9$, the threshold iteration is already thousands of times faster; at $|X| \geq 10^{10}$ the basis method exceeds the time or memory budget, but the threshold iteration completes. Let R_{thr} denote the number of threshold rounds. A threshold round computes one full $\Phi(K)$ update over all columns, whereas a lazy pass incrementally mutates the exposed antichain boundary. Thus, R_{lazy} and R_{thr} are algorithm-specific round counts and are not comparable.

Experiments were run on an NVIDIA RTX 5090 GPU (32 GB, 170 SMs) with an AMD Ryzen 9 9950X3D CPU. We implement the method in C++/pFaces [11] with OpenCL kernels. Each key $k \in \mathcal{K}$ evaluates predecessor feasibility and updates $\tau(k)$ by binary search. Memory is dominated by $\tau \in \mathbb{N}^{|\mathcal{K}|}$ with $|\mathcal{K}| = O(N^{d-1})$.

B. Real-time controller: unprotected left turn

To demonstrate online use of the synthesized invariant sets, we employ the threshold method inside an MPPI [12] controller for an *unprotected left turn* (Figure 2). The ego vehicle approaches the intersection from the west while two oncoming vehicles approach from the east. Their velocities are sensed online. The ego first waits for onc_1 to clear the conflict zone, then commits to the turn before onc_2 closes the available gap. Each phase is formulated as a 3D robust-safety problem in $(s_e, v_e, s_{\text{onc}})$, and the corresponding invariant set is re-synthesized when the sensed oncoming speed changes sufficiently. Let $K_w \subseteq \mathbb{R}^3$ and $K_g \subseteq \mathbb{R}^3$ be the two phase sets. The two-oncoming safety condition is represented by the

¹<https://github.com/mkhaled87/pFaces-MonoSynth>

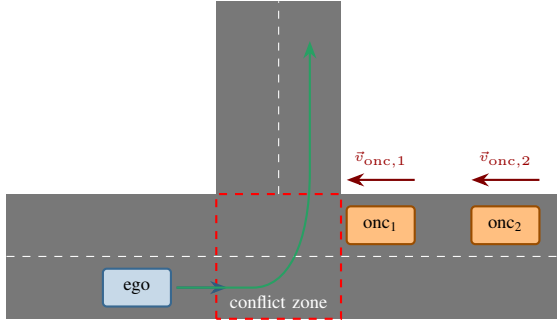


Fig. 2. Real-time controller for an unprotected left turn. An MPPI controller samples rollouts and queries the threshold table once per rollout step to evaluate the set-membership penalty. The resulting optimized control input is applied. When oncoming velocities change, the invariant set is resynthesized.

intersection of these two 3D constraints in lifted coordinates:

$$K_2 = \{(s_e, v_e, s_1, s_2) : (s_e, v_e, s_1) \in K_w, (s_e, v_e, s_2) \in K_g\}.$$

This lifted set is used only for online rollout membership queries; we do not claim that the full 4D system is monotone or that K_2 is a 4D controlled-invariant set. The reduction is sound for simultaneously checking the two phase certificates under the modeled independent oncoming evolution. A formal 4D invariance claim would additionally require a common witness input for both component constraints, which need not exist because the wait and go constructions impose opposite orthant orders on (s_e, v_e) .

The invariant-set algorithm returns K_* , not an explicit policy. In closed loop, we use K_* inside an optimizer: the threshold table gives an $O(1)$ membership query that is easy to insert into a sampling-based MPPI cost. MPPI is a natural choice here because it handles nonlinear dynamics and nonsmooth table penalties without requiring gradients of the threshold representation; first-order MPC methods would need smoothing of the same discontinuous membership test.

Let T_s denote the controller sampling period, H_p the prediction horizon, H_c the control horizon, N_r the number of rollouts per MPPI iteration, N_{it} the number of MPPI updates per control step, σ_u the standard deviation of the Gaussian input perturbations, and $[u_{\min}, u_{\max}]$ the admissible torque interval. Their numerical values are reported in Table III. The stage cost is $\ell(x_k, u_k) = w_p(s_{e,k} - g)^2 + w_s \mathbf{1}_{\{x_k \notin K_2\}} + w_j \|u_k - u_{k-1}\|^2$. Each table is computed for the oncoming-speed parameters and interval vehicle parameters, and re-synthesis is triggered when the sensed velocity change exceeds the threshold ε_{res} listed in Table III. The safety-informed MPPI performs an $O(1)$ threshold lookup per rollout step; states outside K_2 accumulate a penalty w_s , rather than being prohibited by a hard constraint, and the implemented controller provides no formal closed-loop safety guarantee. Such a guarantee would require a hard one-step invariant constraint or an invariant-preserving backup policy, which is left for future work. The threshold lookup and MPPI control computation times are negligible relative to the re-synthesis times.

TABLE III
REAL-TIME UNPROTECTED LEFT TURN: CONFIGURATION AND TIMINGS.

Quantity	Value
Grid cells $ X $	$10^{10} = 10^4 \times 10^2 \times 10^4$
Sampling period T_s	0.1 s
Prediction and control horizon H_p, H_c	20 steps
MPPI iterations / step N_{it}	10
Rollouts / iteration N_r	512
Perturbation standard deviation σ_u	1000 Nm
Cost weights (w_p, w_s, w_j)	$(1, 10^4, 10^{-2})$
Resynthesis threshold ε_{res}	0.2 m/s
Wait and go synthesis $t^{\text{wait}}, t^{\text{go}}$	$77 \pm 15, 62 \pm 7$ ms
Average MPPI control time	2 ms

V. CONCLUSION

We introduced a threshold-function reformulation of greatest-fixed-point invariant-set synthesis for monotone systems. The algorithm is embarrassingly parallel and orders of magnitude faster than lazy fixed-point iteration. Experiments on benchmarks confirm speedups on the order of thousands over the lazy baseline. The threshold iteration reaches grids exceeding 10^9 cells around 100 ms, enabling real-time re-synthesis for closed-loop control. However, the method requires monotonicity under some orthant order and lower-closedness of the safety set. The threshold table stores only the set and gives no specific control policy. Grid resolution is bounded by GPU memory ($O(N^{d-1})$ integers).

REFERENCES

- [1] P. Tabuada, *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media, 2009.
- [2] A. Saoud, E. Ivanova, and A. Girard, “Efficient synthesis for monotone transition systems and directed safety specifications,” in *IEEE Conference on Decision and Control*. IEEE, 2019, pp. 6255–6260.
- [3] E. Ivanova and A. Girard, “Lazy safety controller synthesis with multi-scale adaptive-sampling abstractions of nonlinear systems,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1837–1843, 2020.
- [4] E. Ivanova, A. Saoud, and A. Girard, “Lazy controller synthesis for monotone transition systems and directed safety specifications,” *Automatica*, vol. 135, p. 109993, 2022.
- [5] E. S. Kim, M. Arcak, and S. A. Seshia, “Symbolic control design for monotone systems with directed specifications,” *Automatica*, vol. 83, pp. 10–19, 2017.
- [6] —, “Flexible computational pipelines for robust abstraction-based control synthesis,” in *International Conference on Computer Aided Verification*. Springer, 2019, pp. 591–608.
- [7] A. Saoud and M. Arcak, “Characterization, verification and computation of robust controlled invariants for monotone dynamical systems,” *Mathematics of Control, Signals, and Systems*, vol. 36, pp. 71–100, 2024.
- [8] M. Rungger and M. Zamani, “SCOTS: A tool for the synthesis of symbolic controllers,” in *Proceedings of the 19th international conference on hybrid systems: Computation and control*, 2016, pp. 99–104.
- [9] Y. Li and J. Liu, “ROCS: A robustly complete control synthesis tool for nonlinear dynamical systems,” in *21st International Conference on Hybrid Systems: Computation and Control*, 2018, pp. 130–135.
- [10] S. W. Smith, A. Saoud, and M. Arcak, “Monotonicity-based symbolic control for safety in driving scenarios,” *IEEE Control Systems Letters*, vol. 6, pp. 830–835, 2021.
- [11] M. Khaled and M. Zamani, “pFaces: An acceleration ecosystem for symbolic control,” in *22nd ACM International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 252–257.
- [12] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, “Information theoretic MPC for model-based reinforcement learning,” in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 1714–1721.