LJMU Research Online

Ait El Manssour, R, Kenison, G, Shirmohammadi, M and Varonka, A

 Simple Linear Loops: Algebraic Invariants and Applications

http://researchonline.ljmu.ac.uk/id/eprint/24814/

Article

Simple Linear Loops: Algebraic Invariants and Applications

RIDA AIT EL MANSSOUR, CNRS - IRIF, France
GEORGE KENISON, Liverpool John Moores University, United Kingdom
MAHSA SHIRMOHAMMADI, CNRS - IRIF, France
ANTON VARONKA, TU Wien, Austria

The automatic generation of loop invariants is a fundamental challenge in software verification. While this task is undecidable in general, it is decidable for certain restricted classes of programs. This work focuses on invariant generation for (branching-free) loops with a single linear update.

Our primary contribution is a polynomial-space algorithm that computes the strongest algebraic invariant for simple linear loops, generating all polynomial equations that hold among program variables across all reachable states. The key to achieving our complexity bounds lies in mitigating the blow-up associated with variable elimination and Gröbner basis computation, as seen in prior works. Our procedure runs in polynomial time when the number of program variables is fixed.

We examine various applications of our results on invariant generation, focusing on invariant verification and loop synthesis. The invariant verification problem investigates whether a polynomial ideal defining an algebraic set serves as an invariant for a given linear loop. We show that this problem is **coNP**-complete and lies in **PSPACE** when the input ideal is given in dense or sparse representations, respectively. In the context of loop synthesis, we aim to construct a loop with an infinite set of reachable states that upholds a specified algebraic property as an invariant. The strong synthesis variant of this problem requires the construction of loops for which the given property is the strongest invariant. In terms of hardness, synthesising loops over integers (or rationals) is as hard as Hilbert's Tenth problem (or its analogue over the rationals). When the constants of the output are constrained to bit-bounded rational numbers, we demonstrate that loop synthesis and its strong variant are both decidable in **PSPACE**, and in **NP** when the number of program variables is fixed.

CCS Concepts: • **Theory of computation** → **Logic and verification**; • **Computing methodologies** → **Algebraic algorithms**.

Additional Key Words and Phrases: Algebraic Invariant, Program Synthesis, Loop Invariant, Zariski Closure, Polynomial Space, Algebraic Reasoning.

1 Introduction

Reasoning about loops is a foundational task in program analysis and verification. Loop invariants play a crucial and indispensable role; for instance, they help establish both safety properties (as seen in proofs of non-reachability) and liveness properties (as supporting invariants in termination

---

Authors' Contact Information: Rida Ait El Manssour, CNRS - IRIF, Paris, France, manssour@irif.fr; George Kenison, Liverpool John Moores University, Liverpool, United Kingdom, g.j.kenison@ljmu.ac.uk; Mahsa Shirmohammadi, CNRS - IRIF, Paris, France, mahsa@irif.fr; Anton Varonka, TU Wien, Vienna, Austria, anton.varonka@tuwien.ac.at.

proofs). The paper [14] goes so far as to call the problem of automatic invariant generation *the most important task in program verification.*

The focus of this paper is on the algorithmic generation of algebraic invariants for programs. The invariants we study are given by polynomial equations in the program variables. Not only are these invariants expressive, but they are amenable to a rich collection of techniques from algebraic geometry. From a computational perspective, a key question is to determine when the invariant generation problem is decidable. For decidability, the program model must be fairly abstract, as for instance, algebraic invariant generation is already undecidable for polynomial programs [39].

Müller-Olm and Seidl [61] considered the generation of polynomial invariants within the framework of affine programs, raising the question of whether it is possible to compute (a basis of) all polynomial invariants for any given affine program. This problem can be recast in purely algebraic terms as a question about matrix semigroups: namely, the objective is to compute a representation of the Zariski closure of a finitely generated semigroup of matrices. If the matrices involved are all invertible then this task is equivalent to that of computing the Zariski closure of a finitely generated matrix *group*.

The first algorithm to compute the polynomial ideal defining the Zariski closure $\overline{\langle M_1, \ldots, M_k \rangle}$ of the group generated by a set of invertible matrices $M_1, \ldots, M_k$ was introduced in [28]. This algorithm was recently employed in [39, 40] as a subroutine to address the above-mentioned question posed by Müller-Olm and Seidl regarding invariant generation for affine programs. Both of these decision procedures have significant shortcomings in terms of computational complexity; notably, the complexity bound for the group-closure computation in [28] is not known to be elementary [63, Appendix C].

A recent advance in computing the group closure was obtained through a linearization technique from [46, 60] in combination with a novel upper bound on the degrees of the polynomials defining the closure [63]. The resulting complexity bounds, although elementary, are of the order of severalfold exponential time for rational matrices; nevertheless there still remains a significant gap between the upper and lower complexity bounds.

This paper aims to address and resolve the computational complexity for cyclic matrix groups and semigroups. Surprisingly, the approaches in [28, 39, 40, 63], and other works such as [27, 31], fail to achieve reasonable complexity even in the simplest setting of cyclic matrix groups and semigroups. This complexity blow-up is a consequence of using variable elimination in the computational procedures, which generally has a worst-case exponential-space complexity. Alternative approaches based on reductions to the theory of real-closed fields, combined with the degree upper bound obtained in [63], will also result in similar complexity bounds. In this case, the complexity blow-up arises from the doubly exponential space required for quantifier elimination in real-closed fields [25].

In the vocabulary of loop programs, the above simple setting translates to that of branch-free loops with a single linear update, unlike the commonly studied programs with multiple linear updates [40, 51]. Notwithstanding the radical simplicity of this model, the most natural verification problems (such as termination and reachability) are already very difficult in this setting [64]. We refer to this subclass as the *simple linear loops*, and focus on the invariant generation problem for such loops.

By performing the updates simultaneously, a simple linear loop is specified by a single square matrix M and a vector $\boldsymbol{\alpha}$ of initial program values. The program state is given by a vector, and an iteration of the loop body is summarised by a matrix-vector product. We call a loop with d program variables d-dimensional. The *orbit* of the loop $\langle M, \boldsymbol{\alpha} \rangle$ is the reachable set of program states, defined by $O := \{M^n \boldsymbol{\alpha} : n \in \mathbb{N}\}$. Our main contribution is as follows.

THEOREM 1.1. *Let $\langle M, \boldsymbol{\alpha} \rangle$ be a rational loop with orbit O. A set of polynomials defining $\overline{O}$ is computable in* PSPACE. *For all fixed $d \in \mathbb{N}$, the computation of $\overline{O}$ for d-dimensional loops is in polynomial time.*

Loops with linear updates appear in many settings, such as Markov chains and linear hybrid automata. In [55], the strongest algebraic invariant is generated for a class of linear hybrid automata, where the permitted state transitions are linear updates. Further motivation for complexity results for prototypical classes of loops, as in Theorem 1.1, arises from bottom-up approaches to invariant generation, where one works to summarise larger and larger subprograms in order to analyse expressive program models (cf. [24]).

In general, the strongest algebraic invariant of a loop with linear updates and equality guards is not computable [60]. Similarly, allowing multiple polynomial updates renders invariant generation undecidable [39], or Skolem-hard [62] with a single polynomial update. Thus linear loops are arguably the richest model where the invariant generation problem admits a complete algorithmic procedure.

Applications in Invariant Verification and Loop Synthesis. The concept of orbit closure is fundamental to numerous subfields of computer science, including geometric complexity theory, quantum computation, non-convex optimisation problems, and graph isomorphism [15, 17, 18, 28, 30]. Our problem of invariant generation can be interpreted as an implicit orbit-closure problem under the action of a cyclic semigroup [2].

The first application of invariant generation that we consider is the *invariant verification* problem. In our setting this amounts to checking whether all reachable states of a given loop satisfy a given collection of polynomial equations. We do not assume that the property to be verified is inductive, and so our analysis involves a non-trivial examination of reachability in the loop. The fact that our invariants are equations plays a key role. For example, allowing inequalities in our invariants would render the invariant verification problem more general than the positivity problem for linear recurrence sequences, whose decidability status has been open for many decades [65].

The second application of our algorithm for invariant generation is *loop synthesis*. An example of the kind of scenario we seek to model is as follows. Imagine that a loop has integer variables x, y, w, and z. If, in each iteration of the loop, both w and y are incremented by 1, how should we update variables x and z to preserve the invariant $x^2 - y^2 z^2 + z^3 = 0$? In other words, we ask to synthesise variable updates that maintain a given relation among the loop variables. In the example at hand, one solution is $x := w(y^2 - w^2)$ and $z := y^2 - w^2$. We note that this task is related to the problem of computing a parametrisation of a variety given by polynomial equations, which is a classical problem in algebraic geometry.

We prove that the invariant verification problem is coNP-complete when the polynomials describing the invariant to be verified are given in dense representation. We also show that the problem lies in PSPACE when the polynomials are given in sparse representation. In the context of synthesising loops over integers and rationals, building on our work in invariant generation, we consider bit-bounded variants of the synthesis problem and provide PSPACE algorithm for these cases. Additionally, when the loop dimension is fixed, we establish NP upper bounds and, in some cases, provide matching lower bounds.

A summary of our main results is provided in Fig. 1. A comprehensive discussion of these results, along with related previous works, is presented in Section 2. The full version of this paper, including the appendices, is available at [1].

Invariant Generation (decision problem)

rational loops	rational loops fixed dimension
PSPACE	**PTIME**

Loop Synthesis over $R \in \{\mathbb{Q}, \mathbb{Z}\}$ (decision problem)

weak — as hard as HTP over R	strong — remains open

Bit-Bounded

weak — **PSPACE and NP-hard**	strong — **PSPACE**

Bit-Bounded fixed dimension

weak — **NP** — NP-hard over $\mathbb{Z}$	strong — **NP**

Invariant Verification (decision problem)

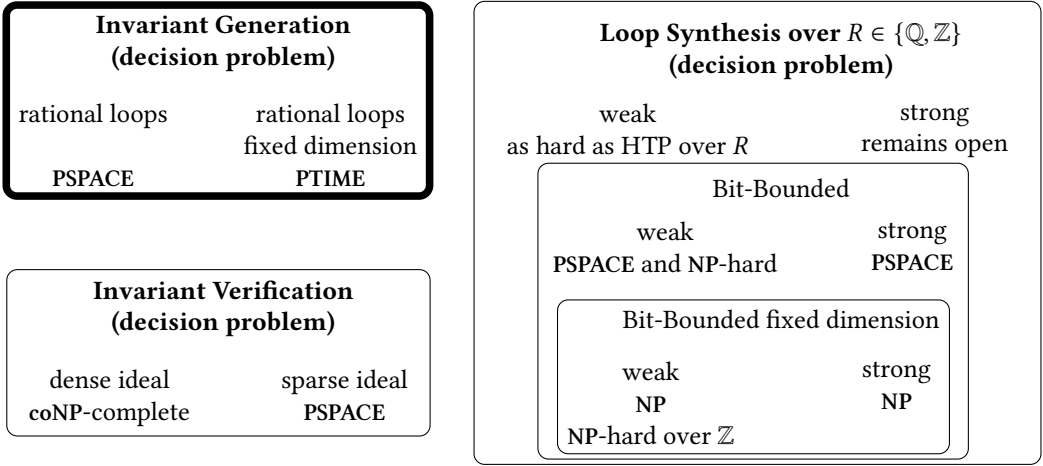| dense ideal	sparse ideal
coNP-complete	**PSPACE**

Fig. 1. Summary of our main results (see Section 2 for a comprehensive overview).

2 Overview of Main Results

In this section, we give a high-level overview of our main results. We will also introduce the basic definitions required to follow the main techniques exhibited in the algorithms we introduce. See Appendix A in [1] for extended preliminaries.

We denote by $\mathbb{Z}$ and $\mathbb{Q}$ the set of all integer and rational numbers, respectively. We write $\mathbb{Q}[x]$ for the ring of univariate polynomials with rational coefficients over x, and $\mathbb{Q}[\boldsymbol{x}]$ for the ring of multivariate polynomials over variables $\boldsymbol{x} = (x_1, \dots, x_d)$. Given a polynomial $P \in \mathbb{Q}[\boldsymbol{x}]$, its total degree is defined as the maximum total degree of its constituent monomials. Following [53], we define $P(\boldsymbol{x})$ as being written in *dense representation* if it is given as an array of coefficients (both zero and nonzero) for all monomials up to its total degree. In contrast, it is in *sparse representation* if it is given as an explicit set of monomials with nonzero coefficients. The size of the polynomial $P(\boldsymbol{x})$, in either representation, is the bit length of a reasonable encoding of the polynomial in the required representation, with all numbers written in binary. For instance, the size of x^{2^n} is $2^n + 1$ when written in dense representation, whereas it is $n + 2$ when written in sparse representation. By the above, the degree of $P(\boldsymbol{x})$ in dense representation is at most its size, whereas in sparse representation it could be exponential in its size. Given a set $S \subseteq \mathbb{Q}[\boldsymbol{x}]$, its description size is defined as the combination of its cardinality, the size of its polynomials, and the number of variables.

Recall that a complex number is algebraic if it is a root of a nonzero univariate polynomial in $\mathbb{Q}[x]$. Denote by $\overline{\mathbb{Q}}$ the set of all algebraic numbers. The *minimal polynomial* of $\alpha \in \overline{\mathbb{Q}}$, denoted by m_α, is uniquely defined as the monic polynomial in $\mathbb{Q}[x]$ of smallest degree for which α is a root. For algorithmic purposes, we rely on a *symbolic representation* of algebraic numbers[1], which results in effective arithmetic, see [22, Section 4.2.1] for more details.

Complexity Theory. We briefly summarise some relevant notions from complexity theory [7]. The Arthur–Merlin complexity class (**AM**) consists of all decision problems that admit a two-round interactive proof in which Arthur tosses some coins and sends the result to Merlin, who responds with a purported proof of membership in the language [8]. It is well-known that **AM** contains both **BPP** (that is, the class of randomised polynomial time with two-sided error) and **NP**. It is also contained in Π_2, the second level of the polynomial hierarchy (**PH**).

---

[1]The symbolic representation of $\alpha \in \overline{\mathbb{Q}}$ consists of m_α, in sparse representation, combined with a triple $(a, b, R) \in \mathbb{Q}^3$ such that α is the unique root of m_α that lies within the R-radius circle centered at (a, b) in the complex plane.

Denote by $\exists\mathbb{R}$ the class of problems that are polynomial-time reducible to the decision problem for the existential theory of the reals. Since the latter lies in **PSPACE** we have that $\exists\mathbb{R} \subseteq$ **PSPACE**. We recall that **PSPACE** is closed under **PSPACE**-oracles, that is, **PSPACE**$^{\text{PSPACE}}$ = **PSPACE**.

Simple Linear Loops. A simple linear loop consists of a single-path loop with linear variable updates. For example, consider the loop in Fig. 2. It is easy to see that after n iterations of the `while` loop, the values of the variables x and y are given by the $(n+1)$th and nth Fibonacci numbers, respectively. After n iterations, the value of variable z is $(-1)^n$.

Input: $(x, y, z) \leftarrow (1, 0, -1)$
  while `true` **do**

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \leftarrow \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

  end while

Fig. 2. Fibonacci simple loop program.

Recall that $\text{Mat}_d(\overline{\mathbb{Q}})$ is the set of all $d \times d$ matrices with entries in $\overline{\mathbb{Q}}$. A *simple linear loop* $\mathcal{L} = \langle M, \boldsymbol{\alpha} \rangle$ is formally defined as a loop program of the form

$$\boldsymbol{x} \leftarrow \boldsymbol{\alpha}; \text{ while } \textbf{true} \text{ do } \boldsymbol{x} \leftarrow M\boldsymbol{x},$$

where $\boldsymbol{\alpha} \in \overline{\mathbb{Q}}^d$ is an *initial vector* and $M \in \text{Mat}_d(\overline{\mathbb{Q}})$ is an *update matrix*. The main focus of this paper is on rational and integer loops, where the constants of M and $\boldsymbol{\alpha}$ lie in $\mathbb{Q}$ and in $\mathbb{Z}$, respectively. We define loops more generally to encompass intermediate steps in our procedures.

The orbit of $\mathcal{L}$ is the reachable set of program states defined by $O := \{M^n \boldsymbol{\alpha} : n \in \mathbb{N}\} \subseteq \overline{\mathbb{Q}}^d$. The orbit of the program in Fig. 2 is

$$\left\{ \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix}, \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 5 \\ 3 \\ -1 \end{pmatrix}, \begin{pmatrix} 8 \\ 5 \\ 1 \end{pmatrix}, \ldots \right\}.$$

A program with a finite orbit is called trivial, and non-trivial otherwise. The above program is non-trivial [50].[2]

Algebraic Geometry. A polynomial ideal I is an additive subgroup of $\mathbb{Q}[\boldsymbol{x}]$ that is closed under multiplication by polynomials in $\mathbb{Q}[\boldsymbol{x}]$. Hilbert's Basis theorem states that every polynomial ideal $I \subseteq \mathbb{Q}[\boldsymbol{x}]$ is *finitely generated*, equivalently, that every strictly ascending chain of ideals in $\mathbb{Q}[\boldsymbol{x}]$ is finite. We shall write $I = \langle S \rangle$ if $S \subseteq \mathbb{Q}[\boldsymbol{x}]$ is a generating set for I.

We view program orbits as a subset of the affine space $\overline{\mathbb{Q}}^d$ for some dimension d. Following [39, 40], we consider over-approximations of program orbits by *algebraic sets*. An algebraic set (or *variety*) is the set of common zeros of a polynomial ideal I; thus, the set $V(I) := \{\boldsymbol{v} \in \overline{\mathbb{Q}}^d : f(\boldsymbol{v}) = 0 \text{ for all } f \in I\}$, is an algebraic set. By Hilbert's Basis theorem, every algebraic set can be represented as the set of common zeros of a finite set of polynomials. An algebraic set $X \subseteq \overline{\mathbb{Q}}^d$ is *irreducible* if it cannot be written as $X = X_1 \cup X_2$ such that X_1 and X_2 are both algebraic sets and both proper subsets of X. In this paper we use the terms algebraic set and variety interchangeably whereas certain authors reserve the term variety for irreducible algebraic sets.

The *strongest algebraic approximation* for a program is the smallest algebraic set containing the program orbit. In this context, *smallest* refers to the closure in the Zariski topology on $\overline{\mathbb{Q}}^d$, where

---

[2]The definition of a non-trivial loop aligns with that of a *wandering point* in the arithmetic dynamics literature [12, 69].

closed sets are algebraic subsets of $\overline{\mathbb{Q}}^d$. For any set $X \subseteq \overline{\mathbb{Q}}^d$, its closure in the Zariski topology on $\overline{\mathbb{Q}}^d$, denoted by $\overline{X}$, is the smallest algebraic set that contains it. Subsequently, the strongest algebraic approximation of a program is $\overline{O}$.

Algebraic Invariant. Given a loop program $\mathcal{L} = \langle M, \boldsymbol{\alpha} \rangle$, a set $A \subseteq \overline{\mathbb{Q}}^d$ is an *invariant* if it over-approximates the orbit O of $\mathcal{L}$, that is, $\{M^n \boldsymbol{\alpha} : n \in \mathbb{N}\} \subseteq A$. Moreover, if A is an algebraic set then we call A an *algebraic invariant*. We may refer to the polynomial ideal $I \subseteq \mathbb{Q}[\boldsymbol{x}]$ such that $V(I) = A$ as an invariant ideal for $\mathcal{L}$.

We say that a set $A \subseteq \overline{\mathbb{Q}}^d$ is *inductive* with respect to $\mathcal{L}$ if $\boldsymbol{\alpha} \in A$ and $\{Mv : v \in A\} \subseteq A$. It is immediate that an inductive set is an invariant. It is equally straightforward that the converse fails in general: not every invariant is inductive. However, the loop $\mathcal{L}$ admits a smallest algebraic invariant, namely the Zariski closure $\overline{O}$ of the orbit, and this set *is* inductive. Indeed $M(\overline{O}) \subseteq \overline{M(O)} \subseteq \overline{O}$ holds by Zariski continuity of the self-map $v \mapsto Mv$ on $\overline{\mathbb{Q}}^d$. A non-trivial algebraic invariant for the program depicted in Fig. 2 is (the zero set of) the polynomial

$$(y^2 + xy - x^2)^2 - z^2 = 0; \tag{1}$$

see, for example [49]. This set is an invariant by virtue of being inductive. Indeed, we have that the initial point $(1, 0, -1)$ satisfies (1). Further, by substitution of the update assignments in Fig. 2, we have that $(x^2 + (x + y)x - (x + y)^2)^2 - (-z)^2 = (y^2 + xy - x^2)^2 - z^2 = 0$. In other words, Equation (1) is stable under the update matrix of the loop.

2.1 Algebraic Invariant Generation

Our main contribution (Theorem 1.1) is a PSPACE algorithm that computes the strongest algebraic invariant of a given simple linear loop. This algorithm runs in polynomial time if the dimension of the loop is fixed.

The geometric properties of the algebraic closure of cyclic subsemigroups of $\mathrm{Mat}_d(\mathbb{Q})$ are studied in [31]. Principally, it is shown that each irreducible component of the closure of a cyclic semigroup is either an isolated point or isomorphic to a toric variety. Recall that a *toric variety* is the closed image of a monomial map. Toric varieties form an important and rich class of varieties in algebraic geometry, particularly in view of their combinatorial and algorithmic properties. To obtain the result in Theorem 1.1, we rely on the observation that the closures of cyclic subsemigroups of matrices and orbit closures of linear loops, under some isomorphism of varieties, have the same geometric structure. A careful analysis of the results in [31], which inspired our research, provides an exponential degree bound for the polynomials defining the closure of matrix semigroups. Such an exponential degree bound, combined with linearization techniques in [46, 60], gives an inefficient EXPSPACE procedure for invariant generation.

In our construction, the main obstacle to improve the PSPACE bound is a basis computation for the lattice of multiplicity relations between the eigenvalues of the update matrix. The best known bound for this task is through a brute-force search in combination with Masser's bound, see [28, 31, 39, 63] and Appendix A in [1]. If the eigenvalues of the matrix updates are rational numbers, rational multiples of roots of unity, or rational multiples of unnested radicals, this task can be performed more efficiently through identity testing problems for the underlying fields [9, 10]. In those cases, this allows us to place the invariant generation problem in the second level of the polynomial hierarchy (PH).

Related Work: Invariant Generation. Automatically generating invariants for simple linear loop programs has garnered significant attention. This task remains highly challenging even for simple loop programs with multiple linear updates [40, 43, 51, 54].

There is also a line of work addressing more expressive computational models, such as programs with nested loops, conditional branching, probabilistic updates, and unstructured control flow. However, these procedures are often limited by the types of polynomial invariants they can generate, rendering them necessarily incomplete. Typically, they produce polynomials only up to a user-defined degree [5, 6, 11, 61]. Amongst recent works, a complete method for generating polynomial invariants for extended P-solvable loops is presented in [44]. Another recent work [24] generates polynomial invariants for programs with arbitrary control flow, building on an earlier heuristic framework in [51]. The approach is complete for solvable transition ideals, a generalisation of solvable polynomial maps [67]. However, the Gröbner basis calculations in [24, Algorithm 3] dominate the running time, making the computation exponential [24, pg.21]. Recently, invariant generation for moment-computable probabilistic loops was shown to be decidable [62]; the proof employs a reduction to invariant generation in our setting (that of simple linear loops).

2.2 Invariant Verification

In this subsection, we consider two decision problems concerning invariants for a given loop. The first problem asks whether a set is an inductive invariant; the second problem asks whether a set is an invariant, not necessarily inductive.

Determining whether an input ideal defines an *inductive set* for a given loop is in principal a simple task, reducible to *radical membership testing*. Recall that the radical of a polynomial ideal $I \subseteq \mathbb{Q}[\mathbf{x}]$, denoted by $\sqrt{I}$, is obtained by taking all roots of its elements within $\mathbb{Q}[\mathbf{x}]$. By Hilbert's Nullstellensatz, if a variety V is such that $V = V(I)$, the only polynomials that vanish on V are those in $\sqrt{I}$. The *radical membership test* asks to determine whether a polynomial $P(\mathbf{x}) \in \mathbb{Q}[\mathbf{x}]$ belongs to the radical of a given ideal $I \subseteq \mathbb{Q}[\mathbf{x}]$ (see [23, Proposition 8, Chapter 4]). Radical membership testing reduces to the satisfiability problem of polynomial equations over $\overline{\mathbb{Q}}$. This is a byproduct of the *Rabinowitsch trick*, see [23, Chapter 4. §2] for example, which shows that $f \in \sqrt{I}$ if and only if $V(\langle I \cup \{1 - yf\}\rangle) = \emptyset$ where $I \subseteq \mathbb{Q}[\mathbf{x}]$ and $\langle I \cup \{1 - yf\}\rangle \in \mathbb{Q}[\mathbf{x}, y]$.

The satisfiability problem for polynomial equations over $\overline{\mathbb{Q}}$, also known as the *Hilbert's Nullstellensatz* problem (HN for short), takes as an input a set S of polynomial equations and asks to determine whether the system is satisfiable, i.e., whether the variety of the ideal $\langle S \rangle$ is non-empty. It is known that the HN problem admits an **AM** protocol [52] under the generalised Riemann hypothesis. This result is independent of the representation of the input polynomials.

By the above, an algorithm to decide whether an input ideal $\langle S \rangle \subseteq \mathbb{Q}[\mathbf{x}]$ defines an inductive set for a given loop $\langle M, \boldsymbol{\alpha} \rangle$ must (1) determine whether $\boldsymbol{\alpha}$ satisfies all relations in S, and (2) test whether $P(M\mathbf{x}) \in \sqrt{\langle S \rangle}$ for all polynomials $P(\mathbf{x}) \in S$. Both of these tests algorithmically reduce to HN. An alternative approach is to write a query in the existential theory of reals; this leads to an unconditional $\exists \mathbb{R}$ upper bound. We note in passing that for a fixed number of variables such queries can be decided in polynomial time.

Determining whether an ideal defines an inductive set for a linear loop is less demanding than determining whether an ideal defines an invariant (not necessarily inductive). We call the latter the *invariant verification* problem. A conceptually simple algorithm for the invariant verification problem is a backward algorithm, used in similar settings in [11, 13, 47, 48], which examines the ascending chain $I_0 \subseteq I_1 \subseteq I_2 \subseteq \cdots$ of ideals where

$$I_0 = \langle S \rangle \quad \text{and} \quad I_i = \langle P(M^j \mathbf{x}) : P \in S, j \leq i \rangle, \tag{2}$$

for all $i \in \mathbb{N}$. By virtue of $\mathbb{Q}[\mathbf{x}]$ being Noetherian, the ascending sequence of nested ideals in (2) stabilises. In our setting, by properties of linear transformations and the well-known fingerprinting procedure for solving Algebraic Circuit Identity Testing (ACIT) [3], we obtain the following.

PROPOSITION 2.1. *Given $S \subseteq \mathbb{Q}[x]$ in dense representation and a loop with orbit O, verifying whether $O \subseteq V(S)$ is coNP-complete.*

The above naïve algorithm results in an inefficient **EXPSPACE** upper bound when the polynomials in S have sparse representation. One of our contributions shows that invariant verification is in **PSPACE**, even if the input polynomials are given in sparse representation. Our procedure also verifies whether $\overline{O} = V(S)$, a task that cannot be performed through the study of the ideal chain in (2).

PROPOSITION 2.2. *Given $S \subseteq \mathbb{Q}[x]$ and a loop with the orbit O, verifying whether $O \subseteq V(S)$ holds is in PSPACE, and is coNP-hard. Similarly, the test $\overline{O} = V(S)$ can be performed in PSPACE. For all fixed $d \in \mathbb{N}$, both problems for d-dimensional loops are decidable in polynomial time.*

Related Work: Invariant Verification. Invariant verification in the setting of matrix semigroups (a non-deterministic model of loop programs) was shown to be decidable by Dräger [29]. While the program model considered in [29] is more expressive than linear loops (which essentially correspond to matrix semigroups with a single generator), the class of invariants is restricted—the invariants therein are given by conjunctions of linear equations.

2.3 Weak and Strong Loop Synthesis

Another application for the computation of strongest invariants, is the complementary view of synthesising linear loops from a given invariant. Most generally, the loop synthesis problem asks, given a polynomial ideal with generating set $S \subseteq \mathbb{Q}[x]$, whether there exists a linear loop with (an infinite) orbit O such that $\overline{O} \subseteq V(S)$. This problem and its variants have recently received much attention [37, 41, 42]; see the discussion in related works below.

A recently defined variant of the loop synthesis problem sharpens the inclusion of $\overline{O}$ in $V(S)$ by requiring equality of these two varieties. We call this variant the *strong synthesis* problem, whereas we may refer to the loop synthesis problem as the *weak synthesis problem*. As an instance, recall the ideal $I := \langle (y^2 + xy - x^2)^2 - z^2 \rangle$ generated by the polynomial in (1). Given I as the input, the loop program in Fig. 2 is a witness for weak synthesis but not for strong synthesis. Indeed, the strongest polynomial invariant for the loop in Fig. 2 is the ideal generated by $J = \langle y^2 + xy - x^2 - z, z^2 - 1 \rangle$ for which I is a strict subset (meaning that $V(J) \subset V(I)$).

We study both weak and strong synthesis for loops over $\mathbb{Z}$ and $\mathbb{Q}$. An informal discussion in [37, Remark 2.8] connects loop synthesis and Diophantine equations. The immediate observation in Lemma 2.3 formalises this connection. The proof of Lemma 2.3 is given in Appendix B of [1].

LEMMA 2.3. *The weak synthesis problem over $\{\mathbb{Z}, \mathbb{Q}\}$ is as hard as Hilbert's Tenth problem (HTP) over $\{\mathbb{Z}, \mathbb{Q}\}$, even in fixed dimension.*

HTP over $\mathbb{Z}$ was shown undecidable by Matiyasevich in 1970 [58], even for polynomial equations with fixed number of variables (as small as 11 variables) [66]. The decidability of HTP over $\mathbb{Q}$ is a long-standing open problem in the theory of Diophantine Equations [68] and in arithmetic geometry [66]. Lemma 2.3 implies undecidability of the weak synthesis of loops over $\mathbb{Z}$. In [26], it is shown that the problem of asking whether a variety has infinitely many integer points is as hard as HTP over $\mathbb{Z}$. The reduction in [26] can be extended naturally to show that the problem of asking whether a variety has infinitely many rational points is also as hard as HTP over $\mathbb{Q}$. The strong synthesis of loops over $\mathbb{Z}$ and $\mathbb{Q}$ has the flavour of this latter problem, and we leave the decidability open.

Bit-bounded Synthesis. The HTP-hardness of the (weak) synthesis problem motivates the following notion of *bit-bounded synthesis*. The bitsize of a rational number $\frac{a}{b}$, with a and b co-prime

integers, is $\log(|a|) + \log(|b|)$. Given a linear loop $\mathcal{L} = \langle M, \boldsymbol{\alpha} \rangle$, we say that it is *B-bounded rational*, if all entries of M and α are rational numbers with bitsize at most B.[3]

The *strong bit-bounded synthesis problem* asks, given a set $S \subseteq \mathbb{Q}[\boldsymbol{x}]$ of polynomials and a bound $B \in \mathbb{N}$, whether there exists some B-bounded pair $\langle M, \boldsymbol{\alpha} \rangle$ with orbit O such that $\overline{O} = V(S)$. Similarly, the *weak bit-bounded synthesis problem* asks whether $\overline{O} \subseteq V(S)$ holds.

We place the bit-bounded variants of loop synthesis, for both strong and weak cases, in PSPACE through the invariant verification problem. We also establish complexity lower bounds for these problems through reductions from 3SAT and Unique SAT.

PROPOSITION 2.4. *The strong and weak bit-bounded synthesis problems over $\{\mathbb{Q}, \mathbb{Z}\}$ lie in* PSPACE. *The weak variant is* NP-*hard; and the strong variant is* NP-*hard under randomised reductions.*

When synthesising loops with a fixed dimension $d \in \mathbb{N}$, we obtain an NP upper bound. For weak synthesis over $\mathbb{Z}$ we show that the problem is complete for the class NP.

PROPOSITION 2.5. *For all fixed $d \in \mathbb{N}$, the strong and weak bit-bounded synthesis problems in dimension d over $\{\mathbb{Q}, \mathbb{Z}\}$ are in* NP. *Moreover, weak bit-bounded synthesis over $\mathbb{Z}$ is* NP-*complete.*

Related Work: Loop Synthesis. Program synthesis, conceived as the problem of generating constraints that relate unknowns and enforce correctness requirements, has received significant attention [4, 36]. Recent works focusing on polynomial invariants [37, 41, 42, 50] have leveraged algebraic techniques to recast the problem of loop synthesis as that of solving an algebraic system of recurrence sequences. The template-based procedure in the work by Humenberger et al. [41, 42] finds the solution to this system of recurrences by solving a polynomial constraint problem; however, we note their solutions result in loops defined over $\overline{\mathbb{Q}}$. A Diophantine approach to loop synthesis is employed in [37]; therein those authors synthesise loops when given a single quadratic equation as an input. Another recent work [50] gave a procedure for synthesising loops for an input binomial ideal.

Matrix completion is the task of completing a partially defined matrix according to a given specification [45]. Perhaps the most notable variant is the *Netflix Problem* [21] (an application in collaborative filtering [34]). In this application the goal is to complete a matrix of movie recommendations so as to minimise the rank (or some proxy thereof such as the nuclear norm). This is reminiscent of template-based approaches towards program synthesis [70]. In the language of loop synthesis, the partial matrix represents an incomplete program fragment and the task is to complete the program so as to guarantee certain desired polynomial invariants.

3 Algebraic Closure of Linear Loops

Our goal in this section is to construct the strongest algebraic invariant of an input loop. Fix a loop $\mathcal{L}$ with update matrix $M \in \mathbb{Q}^{d \times d}$ and initial vector $\boldsymbol{\alpha} \in \mathbb{Q}^d$. We define the size of $\mathcal{L}$ as $d + \ell$, where ℓ is the bitsize of the entries of M and $\boldsymbol{\alpha}$.

As in [28, 31, 39, 63], our first step is to compute the Jordan normal form of the update matrix M via a Jordan decomposition. A Jordan decomposition of M comprises a Jordan matrix J, and a change-of-basis matrix P which satisfy $M = PJP^{-1}$. Recall that the matrix J is such that the only nonzero entries of J are on its diagonal and its superdiagonal, and that the matrix P in the Jordan decomposition is not unique. In Section 3.1, roughly speaking, we show that P can be chosen such that $P^{-1}\boldsymbol{\alpha}$ is a binary vector. This simplifies the computation of the invariant equations, derived in Section 3.2, and our proposed algorithms. We analyse the computational complexity of the algorithm in Section 3.3. Worked examples demonstrating the algorithm are given in Section 4.

---

[3]*B*-boundedness does not place any restrictions on higher powers of M nor the orbit of the associated loop $\mathcal{L}$.

3.1 A Convenient Jordan Block Decomposition

The *Jordan normal form* $J \in \overline{\mathbb{Q}}^{d \times d}$ of M is given by the direct sum of several so-called Jordan blocks $J_1, \ldots, J_s$. Each block J_i is a square matrix of dimension d_i as follows:

$$J_i := \begin{pmatrix} \lambda_i & 1 & & & \\ & \ddots & \ddots & & \\ & & \lambda_i & 1 \\ & & & \lambda_i \end{pmatrix}.$$

The multiset $\{\lambda_1, \ldots, \lambda_s\}$ of algebraic numbers is equal to the multiset of eigenvalues of M. Indeed, for each eigenvalue λ_i the sum of the sizes of all Jordan blocks corresponding to λ_i is its algebraic multiplicity. Recall that J is unique up to the ordering of its Jordan blocks.

A *Jordan decomposition* of M, denoted by (P, J), consists of the associated Jordan normal form J and an invertible change-of-basis matrix $P \in \overline{\mathbb{Q}}^{d \times d}$ for which $M = PJP^{-1}$. According to the s blocks of (P, J), we can partition d-dimensional vectors $\boldsymbol{v}$ into s block vectors $\boldsymbol{v}_i$ such that

$$\boldsymbol{v} := (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s) \tag{3}$$

where each $\boldsymbol{v}_i$ is a vector of size d_i. The *fingerprint* of $\boldsymbol{\alpha}$ with respect to (P, J) is defined as a binary vector $\boldsymbol{\beta} \in \{0, 1\}^d$ where, for each block $\boldsymbol{\beta}_i = (\beta_{i,1}, \ldots, \beta_{i,d_i})$ with $i \in \{1, \ldots, s\}$,

  • the entry $\beta_{i,j}$ is 1 if and only if j is the largest index such that $\alpha_{i,j}$ is nonzero.

We define a Jordan decomposition (P, J) as *convenient* for $\langle M, \boldsymbol{\alpha} \rangle$ if $P^{-1}\boldsymbol{\alpha}$ is the fingerprint of $\boldsymbol{\alpha}$ with respect to the decomposition.

LEMMA 3.1. *For a loop $\langle M, \boldsymbol{\alpha} \rangle$ and any Jordan decomposition (P, J) of M, there exists a matrix $U \in \mathrm{GL}_d(\overline{\mathbb{Q}})$ such that (PU^{-1}, J) is a convenient Jordan decomposition for $\langle M, \boldsymbol{\alpha} \rangle$. The computation of U is in polynomial time in the size of the loop.*

PROOF. Let (P, J) be a Jordan decomposition of M, wherein all algebraic numbers are represented *symbolically*. The computation of (P, J) and P^{-1} can be performed in polynomial time [19, 20].

Write $J = J_1 \oplus \cdots \oplus J_s$ as a direct sum of Jordan blocks. Consider the corresponding block decomposition of the vector $P^{-1}\boldsymbol{\alpha} = \boldsymbol{\alpha}_1 \oplus \cdots \oplus \boldsymbol{\alpha}_s$. We construct a matrix $U = U_1 \oplus \cdots \oplus U_s$ such that $UP^{-1}\boldsymbol{\alpha} \in \{0, 1\}^d$ and U commutes with J. To this end, suppose that $\boldsymbol{\alpha}_i$ is given by $\boldsymbol{\alpha}_i = (\alpha_{1,i}, \ldots, \alpha_{r,i}, 0 \ldots, 0)^\top$ such that $\alpha_{r,i} \neq 0$. We define

$$U_i := \begin{pmatrix} b_r & \cdots & b_1 & \cdots & 0 \\ & \ddots & & \ddots & \vdots \\ & & \ddots & & b_1 \\ & & & \ddots & \vdots \\ & & & & b_r \end{pmatrix}$$

such that each U_i is upper triangular and Toeplitz (meaning that along each diagonal the entries are constant). The entries $b_1, \ldots, b_r$ are chosen such that $U_i \boldsymbol{\alpha}_i = \boldsymbol{e}_r$, where $\boldsymbol{e}_r$ is the standard unit vector with 1 in the rth position. Specifically, $b_r = 1/\alpha_{r,i}$, and the other b_j's are defined one-by-one for $j = r - 1, \ldots, 1$ by back substitution.

By construction we have $UP^{-1}\boldsymbol{\alpha} \in \{0, 1\}^d$. By a well-known property of upper triangular Toeplitz matrices, each U_i commutes with J_i, implying that $JU = UJ$. Define $Q := PU^{-1}$, and observe that $M = PJP^{-1} = QJQ^{-1}$. Moreover, $Q^{-1}\boldsymbol{\alpha} \in \{0, 1\}^d$ holds and $Q^{-1}\boldsymbol{\alpha}$ is the fingerprint of $\boldsymbol{\alpha}$. Hence, the Jordan decomposition (PU^{-1}, J) is convenient for $\langle M, \boldsymbol{\alpha} \rangle$. □

3.2 Computing the Strongest Algebraic Invariant

Let (P, J) be an arbitrary Jordan decomposition of M such that all Jordan blocks with zero eigenvalue appear first. Consider the orbit $O := \{M^n \boldsymbol{\alpha} : n \in \mathbb{N}\}$ of the loop. We are interested in computing the polynomials that define the algebraic set $\overline{O} = \overline{\{M^n \boldsymbol{\alpha} : n \in \mathbb{N}\}}$. Since $M^n \boldsymbol{\alpha} = PJ^n P^{-1} \boldsymbol{\alpha}$ holds for all $n \in \mathbb{N}$, we compute the polynomials that define $\overline{O}$ in two steps:

- first, we can compute the polynomials that define the algebraic set $\overline{\{J^n (P^{-1}\boldsymbol{\alpha}) : n \in \mathbb{N}\}}$ and
- second, we apply the invertible linear transformation P.

For the decomposition (P, J) of M, let N be the direct sum of all Jordan blocks associated with eigenvalue 0. Let d_0 be the dimension of N; that is, the algebraic multiplicity of 0. Write $J_1, \ldots, J_s$ for all Jordan blocks with associated nonzero eigenvalues $\lambda_1, \ldots, \lambda_s$, respectively. Denote by d_i the dimension of the Jordan blocks J_i. Thus $J = N \oplus J_1 \oplus \cdots \oplus J_s$.

The above-mentioned matrix N is *nilpotent*, i.e., there is an integer $m \leq d_0$ such that $N^m = 0$. Thus the set $\{N^n : n \in \mathbb{N}\}$ is finite, which in turn implies that the algebraic set $\overline{\{J^n (P^{-1}\boldsymbol{\alpha}) : n \in \mathbb{N}\}}$ contains at most m isolated points:

$$\{P^{-1}\boldsymbol{\alpha}, JP^{-1}\boldsymbol{\alpha}, \ldots, J^{m-1}P^{-1}\boldsymbol{\alpha}\}. \tag{4}$$

Write n_0 for the number of distinct isolated points in the above set. Define $\boldsymbol{\gamma} := J^{n_0} (P^{-1}\boldsymbol{\alpha})$. We obtain $\tilde{J}$ from $J = N \oplus J_1 \oplus \cdots \oplus J_s$ by replacing N with the zero matrix (of size $d_0 \times d_0$). By (4) and the definitions of $\boldsymbol{\gamma}$ and $\tilde{J}$, the algebraic set $\overline{\{J^n(P^{-1}\boldsymbol{\alpha}) : n \in \mathbb{N}\}}$ decomposes into

$$\{J^i P^{-1}\boldsymbol{\alpha} \ : \ 0 \leq i < n_0\} \cup \overline{\{\tilde{J}^n \boldsymbol{\gamma} : n \in \mathbb{N}\}}.$$

This allows us to first focus on the Zariski closure of the loop with the invertible transition matrix $J_1 \oplus \cdots \oplus J_s$, and then recover $\overline{O}$.

Towards this goal, we apply Lemma 3.1 to $\langle \tilde{J}, \boldsymbol{\gamma} \rangle$ in order to compute the matrix U for which $(PU^{-1}, \tilde{J})$ is a convenient Jordan decomposition for $\langle \tilde{J}, \boldsymbol{\gamma} \rangle$, which by construction respects the ordering of the Jordan blocks in $\tilde{J}$. Define $\boldsymbol{\beta} := U\boldsymbol{\gamma}$. Consider the partition of $\boldsymbol{\beta}$ according to the decomposition (P, J), defined in (3). Denote by $\boldsymbol{\beta}_i$ the block of $\boldsymbol{\beta}$ that corresponds to the Jordan block J_i. Thanks to the convenient Jordan decomposition, each block $\boldsymbol{\beta}_i$ is either a zero vector, or a standard unit vector. For block $\boldsymbol{\beta}_i$, we refer to k_i as the index of the nonzero entry β_{i,k_i} of $\boldsymbol{\beta}_i$. (We assume $k_i = 0$ in the case that $\boldsymbol{\beta}_i$ is a zero vector.)

Consider $i \in \{1, \ldots, s\}$ such that the block $\boldsymbol{\beta}_i$ is a standard unit vector. By analysing $J_i^n \boldsymbol{\beta}_i$, we find that

$$J_i^n \boldsymbol{\beta}_i = \begin{pmatrix} \binom{n}{k_i-1}\lambda_i^{n-k_i+1} \\ \binom{n}{k_i-2}\lambda_i^{n-k_i+2} \\ \vdots \\ n\lambda_i^{n-1} \\ \lambda_i^n \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{5}$$

holds for all $n \in \mathbb{N}$.

At the next step we introduce a linear transformation $R := I_{d_0} \oplus R_1 \oplus \cdots \oplus R_s$ to simplify (5) such that

$$
R_i J_i^n \boldsymbol{\beta}_i = \begin{pmatrix} n^{k_i-1}\lambda_i^n \\ n^{k_i-2}\lambda_i^n \\ \vdots \\ n\lambda_i^n \\ \lambda_i^n \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{6}
$$

Recall the combinatorial identity

$$
n^k = \sum_{i=1}^{k} i! \begin{Bmatrix} k \\ i \end{Bmatrix} \cdot \binom{n}{i} = \sum_{i=1}^{k} c_{k,i} \binom{n}{i}, \tag{7}
$$

where $\begin{Bmatrix} k \\ i \end{Bmatrix} = \frac{c_{k,i}}{i!}$ is the Stirling number of the second kind. Since the Stirling numbers are defined recursively by the relation $c_{k+1,i} = i(c_{k,i} + c_{k,i-1})$, we can compute the coefficients in R_i, starting with $c_{1,1} = \cdots = c_{k_i-1,1} = 1$, in polynomial time. The block matrix R_i is defined below.

$$
R_i := \begin{pmatrix} c_{k_i-1,k_i-1}\lambda_i^{k_i-1} & c_{k_i-1,k_i-2}\lambda_i^{k_i-2} & \cdots & c_{k_i-1,1}\lambda_i & 0 & \\ & c_{k_i-2,k_i-2}\lambda_i^{k_i-2} & & c_{k_i-2,1}\lambda_i & \vdots & \\ & & & & \vdots & \\ & & \ddots & & & \\ & & & c_{2,2}\lambda_i^2 & c_{2,1}\lambda_i & \\ & & & & c_{1,1}\lambda_i & 0 \\ & & & & & 1 \\ \hline & & & & & & I_{d_i-k_i} \end{pmatrix}. \tag{8}
$$

One can see that (6) is a direct consequence of (5), (7) and (8). We note that, by construction, R is an invertible matrix.

Let $\boldsymbol{x} = (x_1, \ldots, x_d)$ where d is the dimension of the matrix M. Consider the partition of $\boldsymbol{x} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_s)$ according to the decomposition (P, J), defined in (3). Let $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,d_i})$ be the block vector corresponding to the Jordan block J_i with $i \in \{1, \ldots, s\}$ and $\boldsymbol{x}_0 = (x_{0,1}, \ldots, x_{0,d_0})$ the concatenation of the block vectors corresponding to Jordan blocks with zero eigenvalues.

In the following, we construct a set of polynomials in $\mathbb{Q}[\boldsymbol{x}]$ that define the variety

$$
R \cdot \overline{\{\tilde{J}^n \boldsymbol{\beta} : n \in \mathbb{N}\}}, \tag{9}
$$

which in turn helps us define the set of polynomials for the algebraic closure of the orbit of $\langle M, \boldsymbol{\alpha} \rangle$.

Recall that the only nonzero entry of the block $\boldsymbol{\beta}_i$ is indexed by k_i if $\boldsymbol{\beta}_i$ is a standard unit vector, and $k_i = 0$ otherwise. Define the set S_1 of polynomials in $\mathbb{Q}[x_{1,k_1}, \ldots, x_{s,k_s}]$ such that

$$
V(S_1) = \overline{\{(\lambda_1^n, \ldots, \lambda_i^n, \ldots, \lambda_s^n) : n \in \mathbb{N}; \boldsymbol{\beta}_i \neq 0\}}. \tag{10}
$$

The task of computing the defining polynomials is well-understood, see for example [28, 31, 39, 63] and Appendix A in [1] for details. The underlying idea is to compute a basis for the lattice of multiplicity relations between the eigenvalues.

For each Jordan block J_i with $i \in \{1, \ldots, s\}$, define the set $S_{2,i}$ of polynomials in $\mathbb{Q}[x_i]$ as follows:

$$S_{2,i} := \begin{cases} \{x_{i,k_i-1}^j - x_{i,k_i-j} x_{i,k_i}^{j-1} : 2 \le j \le k_i - 1\} & \text{if } k_i \ge 3, \\ \emptyset & \text{otherwise.} \end{cases}$$

This set captures the relations between the nonzero entries of the ith block, as in (6).

For each pair of distinct Jordan blocks J_i, J_j, with $i, j \in \{1, \ldots, s\}$, we define the set $S_{3,i,j}$ of polynomials in $\mathbb{Q}[x_{i,k_i-1}, x_{i,k_i}, x_{j,k_j-1}, x_{j,k_j}]$ as follows:

$$S_{3,i,j} := \begin{cases} \{x_{i,k_i-1} x_{j,k_j} - x_{j,k_j-1} x_{i,k_i}\} & \text{if } k_i \ge 2 \text{ and } k_j \ge 2, \\ \emptyset & \text{otherwise.} \end{cases}$$

This set defines the relations between the two blocks J_i and J_j. Such a relation only exists when in both blocks $\boldsymbol{\beta}_i$ and $\boldsymbol{\beta}_j$, the indices k_i and k_j corresponding to the nonzero entries are both greater than or equal 2.

Define the set S_4 of polynomials in $\mathbb{Q}[x]$ such that

$$S_4 := \{x_{0,j} : 1 \le j \le d_0\} \cup \{x_{i,j} : 1 \le i \le s, k_i < j \le d_i\}$$

captures the zero entries. Define $I_R := \langle S_1, S_{2,i}, S_{3,i,j}, S_4 : i, j \in \{1, \ldots, s\}\rangle$. We are now in the position to prove the following:

CLAIM 3.2. $V(I_R) = R \cdot \overline{\{\tilde{J}^n \boldsymbol{\beta} : n \in \mathbb{N}\}}$.

PROOF. Let $\boldsymbol{v}_n$ be the vector obtained from $\tilde{J}^n \boldsymbol{\gamma}$ under the transformation R, shown explicitly in (6). We consider the block decomposition of $\boldsymbol{v}_n$ according to (P, J).

We did not precisely define the set of polynomials in S_1 as we borrow the technology developed in [28, Lemma 6] to construct S_1 defining the variety in (10). The polynomials in S_1 are defined over the variables $x_{1,k_1}, \ldots, x_{s,k_s}$. This choice is justified by the fact that in the block corresponding to the Jordan block J_i of $\boldsymbol{v}_n$ the k_ith entry is λ_i^n.

For each Jordan block J_i, the set $S_{2,i}$ reflects the relations between corresponding entries in $\boldsymbol{v}_n$. Inspecting (6) again for the entries of $\boldsymbol{v}_n$, the following identity

$$\left(n\lambda_i^n\right)^j - n^j \lambda_i^n \left(\lambda_i^n\right)^{j-1} = 0$$

is realised by $x_{i,k_i-1}^j - x_{i,k_i-j} \cdot x_{i,k_i}^{j-1}$, with $j \in \{2, \ldots, k_i - 1\}$.

Furthermore, for each pair of distinct Jordan blocks J_i, J_j, with $i, j \in \{1, \ldots, s\}$, the set $S_{3,i,j}$ encompasses the relations between the two blocks. As mentioned above, there is no relation between J_i and J_j if either $k_i < 2$ or $k_j < 2$. Otherwise, the identity

$$(n\lambda_i^n)(\lambda_j^n) - (n\lambda_j^n)(\lambda_i^n) = 0$$

is realised by $x_{i,k_i-1} x_{j,k_j} - x_{j,k_j-1} x_{i,k_i}$. The set S_4 reflects the entries in $\boldsymbol{v}_n$ that are identically zero. □

In [31] it was shown that each irreducible component of the variety defined by a cyclic matrix semigroup is isomorphic to the Cartesian product of a toric variety and a normal rational curve (excluding a number of isolated points). Our contribution in this regard is to show an explicit construction of that isomorphism (in the case of loops) can be computed in polynomial time using the matrices R and U. In (6) we read-off the effect of the Cartesian product of a toric variety and a normal rational curve.

It remains to define a generating set of polynomials for $\overline{O}$ from the ideal I_R. From the steps detailed in the construction, we observe that

$$\overline{\{M^n \boldsymbol{\alpha} : n \in \mathbb{N}\}} = PU^{-1}R^{-1}R \cdot \overline{\{\tilde{J}^n \boldsymbol{\beta} : n \in \mathbb{N}\}} \cup \{M^i \boldsymbol{\alpha} : 0 \le i < n_0\}. \tag{11}$$

By this equality, to obtain the polynomials defining $\overline{O}$,

- we first compute ideals $\mathcal{I}_0, \ldots, \mathcal{I}_{n_0-1}$, each defining one of the $n_0 \leq d$ isolated points in $\{M^i \boldsymbol{\alpha} : 0 \leq i < n_0\}$,
- next we obtain an ideal $\mathcal{J}$ by applying the transformation $PU^{-1}R^{-1}$ to the vector of variables $\boldsymbol{x}$ for each polynomial in the set

$$S := (S_1 \cup S_4) \cup \bigcup_{\substack{i,j \in \{1,\ldots,s\} \\ i \neq j}} (S_{2,i} \cup S_{3,i,j}) \tag{12}$$

defining $V(I_R)$. This completes the construction of the strongest algebraic invariant of the loop $\langle M, \boldsymbol{\alpha} \rangle$ as it is simply $V(\mathcal{I}_0 \cap \cdots \cap \mathcal{I}_{n_0-1} \cap \mathcal{J})$. For complexity purposes, in particular to avoid an exponential blow-up, we choose to output the generators of the $\mathcal{I}_i$ and $\mathcal{J}$ separately. (This choice is crucial for the obtained PSPACE bound in Proposition 2.2.)

The obtained polynomials are not in $\mathbb{Q}[\boldsymbol{x}]$ but in $k[\boldsymbol{x}]$ where $k = \mathbb{Q}(\lambda_1, \ldots, \lambda_s)$ is the number field obtained by adjoining the eigenvalues to $\mathbb{Q}$. Furthermore, the eigenvalues are given in symbolic presentation. As an extra step (not necessary to obtain the complexity bound) we convert the coefficient of polynomials to integers. We introduce the variables y_i, one for each eigenvalue λ_i. Recall that each λ_i appears with all its Galois conjugates, and its degree is bounded from above by d. For brevity, given a Galois automorphism σ of the field $\mathbb{Q}(\lambda_i)$ we denote by $\sigma(y_i)$ the variable among $y_1, \ldots, y_s$ corresponding to $\sigma(\lambda_i)$. For each λ_i, write $m_{\lambda_i} = \sum_{k=0}^{n} a_k x^k$ for the minimal polynomial. By basic algebra, we know that the set of equations

$$
\begin{aligned}
a_{n-1} &= - \sum_{\sigma \in \mathrm{Gal}(\mathbb{Q}(\lambda_i)/\mathbb{Q})} \sigma(y_i) \\
a_{n-2} &= \sum_{\sigma,\sigma' \in \mathrm{Gal}(\mathbb{Q}(\lambda_i)/\mathbb{Q})} \sigma(y_i)\, \sigma'(y_i) \\
&\ \ \vdots \\
a_0 &= (-1)^n \prod_{\sigma \in \mathrm{Gal}(\mathbb{Q}(\lambda_i)/\mathbb{Q})} \sigma(y_i)
\end{aligned}
\tag{13}
$$

is such that the solutions to variables $\sigma(y_i)$ define the set of all conjugates of λ_i.

Define the rewrite rules $\tau := \{\lambda_i \to y_i, x_{i,k_i} \to y_i \mid 1 \leq i \leq s\}$. Similarly to the above, we define the ideal $\mathcal{Y} \in \mathbb{Q}[\boldsymbol{x}, y_1, \ldots, y_s]$ obtained by applying the transformation $\tau(PU^{-1}R^{-1})$ to the vector of variables $\boldsymbol{x}$ for each polynomial in (12), the set of polynomials given in (13) for each λ_i, the set $\tau(S_1)$ of multiplicity relations expressed in y_i, together with $\tau(PJP^{-1}) - M$ and $\tau(UP^{-1})\boldsymbol{\alpha} - \boldsymbol{\beta}$.

Observe that $V(\mathcal{J}) = \pi(V(\mathcal{Y}))$ where π is the projection map onto the x_i coordinates, see Lemma B.1 in [1, Appendix B] for a detailed proof.

3.3 Computational Complexity

We now proceed with the statement and proof of our main result.

THEOREM 1.1. *Let $\langle M, \boldsymbol{\alpha} \rangle$ be a rational loop with orbit O. A set of polynomials defining $\overline{O}$ is computable in PSPACE. For all fixed $d \in \mathbb{N}$, the computation of $\overline{O}$ for d-dimensional loops is in polynomial time.*

PROOF. The construction detailed in Section 3.2 can be performed in polynomial time, modulo the computation of the set S_1 of polynomials defining $\overline{\{(\lambda_1^n, \ldots, \lambda_i^n, \ldots, \lambda_s^n) : n \in \mathbb{N}; \boldsymbol{\beta}_i \neq 0\}}$ in (10).

Indeed, every *rational* square matrix has a Jordan decomposition that can be computed in polynomial time [19], and in polynomial time we can shuffle the decomposition so that the Jordan

blocks with zero eigenvalues appear first. Due to the symbolic representation of eigenvalues [22, Section 4.2.1], all constructions in the remaining steps are performed in polynomial time. This includes the computation of U obtained by applying Lemma 3.1 and the computation of its inverse.

It remains to discuss the computation of S_1. Here, we employ the architecture developed in [28, Lemma 6]. Therein, the problem of computing the polynomial ideal that defines the closure $\overline{\{(\lambda_1^n, \ldots, \lambda_i^n, \ldots, \lambda_s^n) : n \in \mathbb{N}; \boldsymbol{\beta}_i \neq 0\}}$ reduces to finding a set of generators for the associated lattice L. To obtain the complexity bounds, we rely on a theorem of Masser [57] that gives an explicit upper bound on the magnitude of the components of a basis for L.

Following [38], membership of a tuple $\boldsymbol{v} \in L$ is in $\exists \mathbb{R}$, using a decision procedure for the existential theory of the reals. In combination with Masser's bound, it follows that we can compute a basis for L in **PSPACE** by brute-force search if the dimension is not fixed, and in polynomial time when the dimension is fixed. See Appendix A in [1] for further details. $\qquad\square$

In the construction above, the input matrix-vector pair $\langle M, \boldsymbol{\alpha} \rangle$ encodes a loop with rational constants. Our procedures naturally extend to cases where the loop constants are algebraic numbers. This is because algebraic numbers can be expressed in the power basis of their splitting field, with each number represented as a vector of rational coefficients. This encoding increases the loop's dimension by a factor equal to the degree of the splitting field over $\mathbb{Q}$. For further details about computations with algebraic numbers see [59].

4 Worked Examples

We illustrate the multi-step procedure of Section 3 with three worked examples. The computations involved in preparing these examples were performed in Macaulay2 [35].

The orbit closure of the loop in Example 4.1 has three isolated points. By design, this example highlights the role of the convenient Jordan form and the transformation R in computing the ideal defining the invariant. As the update matrix in this example has only a single nonzero eigenvalue, the lattice of multiplicity relations is not relevant.

Example 4.1. We run our procedure for generating the strongest algebraic invariant for the loop below.

Input: $(x_1, x_2, x_3, x_4, x_5, x_6) \leftarrow \boxed{\frac{1}{16}(0, 8, 14, -5, 0, 0)}$

    while `true` **do**

$$
\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} \leftarrow \frac{1}{14} \begin{pmatrix} 42 & 0 & -7 & -42 & 21 & 28 \\ -50 & 10 & 0 & 2 & -2 & -20 \\ -26 & -20 & 28 & 52 & -10 & -30 \\ -4 & -2 & 0 & -6 & 6 & 4 \\ 14 & 0 & -14 & -28 & 28 & 14 \\ -38 & -12 & 14 & 48 & -20 & -18 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix}
$$

    end while

Loop 1. Loop defined by the tuple $\langle M, \boldsymbol{\alpha} \rangle$

The tuple (P, J) is a Jordan decomposition of the update matrix M, where

$$
P = \begin{pmatrix}
0 & -1 & 0 & 0 & -1 & -1 \\
2 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & -1 & -1 & -1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & -1 & -1 \\
1 & 1 & 0 & -1 & 1 & 1
\end{pmatrix}
\quad \text{and} \quad
J = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 1 & 0 \\
0 & 0 & 0 & 0 & 2 & 1 \\
0 & 0 & 0 & 0 & 0 & 2
\end{pmatrix}.
$$

The matrix J decomposes into two blocks $J = N \oplus J_1$, where the block N is the nilpotent block associated with the eigenvalue 0 and J_1 is the block associated with the eigenvalue 2. Since $N^3 = 0$, we deduce that $\overline{O}$ has 3 isolated points $\{\boldsymbol{\alpha}, M\boldsymbol{\alpha}, M^2\boldsymbol{\alpha}\}$, see equation (4). Let

$$
\boldsymbol{\gamma} = J^3 P^{-1} \boldsymbol{\alpha} = (0, 0, 0, 0, 0, 1)^{\top} \qquad \text{and} \qquad \tilde{J} = \mathbf{0}_{3\times3} \oplus J_1.
$$

This implies that $(\mathrm{Id}_6, \tilde{J})$ is a convenient Jordan decomposition for $\langle \tilde{J}, \boldsymbol{\gamma} \rangle$; as in Lemma 3.1 and the discussion at (4). For every $n \geq 3$, as in equation (5), we have

$$
M^n \boldsymbol{\alpha} = P\tilde{J}^n \boldsymbol{\gamma} = P \begin{pmatrix}
0 \\
0 \\
0 \\
\frac{n(n-1)}{2} 2^{n-2} \\
n 2^{n-1} \\
2^n
\end{pmatrix}.
$$

Let $R = \mathrm{Id}_3 \oplus \begin{pmatrix} 8 & 2 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. Then for every $n \geq 3$, as in (6), we get

$$
M^n \boldsymbol{\alpha} = PR^{-1} \begin{pmatrix}
0 \\
0 \\
0 \\
n^2 2^n \\
n 2^n \\
2^n
\end{pmatrix}.
\tag{14}
$$

In this simple example, our procedure outputs $S_{2,1} = \{x_4 x_6 - x_5^2\}$ and $S_4 = \{x_1, x_2, x_3\}$. One can also infer these polynomial relations from (14). Altogether, the ideal $\langle x_1, x_2, x_3, x_4 x_6 - x_5^2 \rangle$ defines the closure of $RP^{-1}\{M^n \boldsymbol{\alpha} : n \geq 3\}$. After applying the transformation PR^{-1}, we obtain the polynomials that define the closure of $\{M^n \boldsymbol{\alpha} : n \geq 3\}$. Indeed, the transformed relations generate the polynomial

$$
I = \langle x_5 + x_6, x_4, 2x_1 + x_2 + x_6, 4x_2^2 + 3x_2 x_3 - 3x_3^2 - 3x_2 x_6 + x_3 x_6 - 2x_6^2 \rangle.
$$

It follows that the variety defining the orbit closure $\overline{O}$ of the loop $\langle M, \boldsymbol{\alpha} \rangle$ is given by $V(I) \cup \{\boldsymbol{\alpha}, M\boldsymbol{\alpha}, M^2\boldsymbol{\alpha}\}$. ◄

In the following example we consider a loop whose update matrix has several nonzero eigenvalues. Thus to compute the orbit closure of the loop, our procedure requires us to first compute a basis for lattice of multiplicity relations between the eigenvalues.

Example 4.2. We run our procedure on the loop $\langle M, \boldsymbol{\alpha} \rangle$, given below. The eigenvalues of M are all primitive third and fourth roots of unity.

Input: $(x_1, x_2, x_3, x_4, x_5, x_6) \leftarrow \boxed{(2,\text{-}1,1,1,0,\text{-}1)}$
    while true **do**

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} \leftarrow \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 & 0 & -4 \\ 0 & 0 & 1 & 0 & 0 & -4 \\ 0 & 0 & 0 & 1 & 0 & -4 \\ 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix}$$

end while

Loop 2. Loop defined by the tuple $\langle M, \boldsymbol{\alpha} \rangle$

Since all the eigenvalues of update matrix M are nonzero, there are no isolated points in the orbit closure of the loop. Fix $\omega := -\frac{1}{2} + \frac{\sqrt{3}}{2}\mathrm{i}$. The tuple (P, J) is a Jordan decomposition of M, where

$$P = \begin{pmatrix} -\mathrm{i} & \mathrm{i} & -\omega & \omega^2 & -\omega^2 & \omega \\ 1-2\mathrm{i} & 1+2\mathrm{i} & \omega^2+2 & -2\omega & \omega+2 & -2\omega^2 \\ 2-3\mathrm{i} & 2+3\mathrm{i} & 2\omega^2+3 & \omega^2+1 & 2\omega+3 & \omega+1 \\ 3-2\mathrm{i} & 3+2\mathrm{i} & \omega^2+3 & -2\omega & \omega+3 & -2\omega^2 \\ 2-\mathrm{i} & 2+\mathrm{i} & \omega^2+2 & 1 & \omega+2 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} \quad \text{and} \quad J = \begin{pmatrix} -\mathrm{i} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathrm{i} & 0 & 0 & 0 & 0 \\ 0 & 0 & \omega^2 & 1 & 0 & 0 \\ 0 & 0 & 0 & \omega^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \omega & 1 \\ 0 & 0 & 0 & 0 & 0 & \omega \end{pmatrix}.$$

The spectrum of J (and hence M) is $\{\pm\mathrm{i}, \omega, \omega^2\}$. We next transform our decomposition into a convenient one with a change-of-basis matrix (following Lemma 3.1). We compute an invertible matrix U for which $UP^{-1}\boldsymbol{\alpha} = (1, 1, 0, 1, 0, 1)^{\mathsf{T}}$. Recall that U admits a block diagonal decomposition

$$U := U_1 \oplus U_2 \oplus U_3 \oplus U_4$$

where blocks U_1 and U_2 are size-1 and blocks U_3 and U_4 are size-2. For this computation, we pre-compute the inverse matrix

$$P^{-1} = \frac{1}{18} \begin{pmatrix} -9\mathrm{i} & -9 & 9\mathrm{i} & 9 & -9\mathrm{i} & -9 \\ 9\mathrm{i} & -9 & -9\mathrm{i} & 9 & 9\mathrm{i} & -9 \\ 8(\omega-\omega^2) & 8-2\omega & 4(\omega^2-\omega) & 2\omega^2-8 & 8(\omega-\omega^2) & 14-8\omega \\ 6\omega & 6 & 6\omega^2 & 6\omega & 6 & 6\omega^2 \\ 8(\omega^2-\omega) & 8-2\omega^2 & 4(\omega-\omega^2) & 2\omega-8 & 8(\omega^2-\omega) & 14-8\omega^2 \\ 6\omega^2 & 6 & 6\omega & 6\omega^2 & 6 & 6\omega \end{pmatrix}$$

and the vector $P^{-1}\boldsymbol{\alpha} = \alpha_1 \oplus \alpha_2 \oplus \boldsymbol{\alpha}_3 \oplus \boldsymbol{\alpha}_4$ where

$$\alpha_1 = \frac{3}{2} - \frac{\mathrm{i}}{2}, \qquad\qquad\qquad \alpha_2 = \frac{3}{2} + \frac{\mathrm{i}}{2},$$

$$\boldsymbol{\alpha}_3 = \frac{1}{9}\left(16\omega - 10, 9\omega - 3\right), \text{ and} \qquad \boldsymbol{\alpha}_4 = \frac{1}{9}\left(16\omega^2 - 10, 9\omega^2 - 3\right).$$

By solving the system of linear equations $UP^{-1}\boldsymbol{\alpha} = (1, 1, 0, 1, 0, 1)^{\mathsf{T}}$ in terms of the entries of U, we obtain $U_i = \alpha_i^{-1}$ for $i \in \{1, 2\}$, and

$$U_3 = \begin{pmatrix} \frac{3}{3\omega-1} & \frac{10-16\omega}{(3\omega-1)^2} \\ 0 & \frac{3}{3\omega-1} \end{pmatrix}, \quad \text{and} \quad U_4 = \begin{pmatrix} \frac{3}{3\omega^2-1} & \frac{10-16\omega^2}{(3\omega^2-1)^2} \\ 0 & \frac{3}{3\omega^2-1} \end{pmatrix},$$

which completes the computation of the block matrix U. From Lemma 3.1, we conclude that $\left(PU^{-1}\boldsymbol{\alpha}, J\right)$ is a convenient Jordan decomposition for $\langle M, \boldsymbol{\alpha} \rangle$.

Let $\boldsymbol{\beta} := UP^{-1}\boldsymbol{\alpha}$. The next step is to compute a block matrix $R = R_1 \oplus R_2 \oplus R_3 \oplus R_4$ such that the blocks of $RJ^n\boldsymbol{\beta}$ have the form presented in (6). Since each R_i has size at most 2, the matrix R is diagonal

(see the general form given in (8)). Observe that $R = \text{diag}(1, 1, \omega^2, 1, \omega, 1)$. We now construct a generating set of the ideal I_R defining the variety $V(I_R) = R \cdot \overline{\{J^n \boldsymbol{\beta} : n \in \mathbb{N}\}}$, as described in (9). We consider each of the sets $S_1, S_2, S_3,$ and S_4 of possible polynomial relations amongst the variables in turn.

First, we consider the set S_1 of multiplicative relations between the eigenvalues. This step constructs the exponent lattice $L_{\exp}$ of $\{-i, i, \omega^2, \omega\}$ defined by

$$L_{\exp} = \left\{ (a, b, c, d) \in \mathbb{Z}^4 : (-i)^a i^b (\omega^2)^c \omega^d = 1 \right\}.$$

We invoke a standard subroutine to compute a basis for this lattice (see Appendix A in [1] for the details). An example of such a basis is $L_{\exp}(-i, i, \omega^2, \omega)$ is $\{(4, 0, 0, 0), (1, 1, 0, 0), (0, 0, 3, 0), (0, 0, 1, 1)\}$ and so we deduce that the polynomials

$$p_{11}(\boldsymbol{x}) = x_1^4 - 1, \quad p_{12}(\boldsymbol{x}) = x_1 x_2 - 1, \quad p_{13}(\boldsymbol{x}) = x_4^3 - 1, \quad p_{14}(\boldsymbol{x}) = x_4 x_6 - 1$$

define the set S_1.

We next consider each of the sets $S_{2,i}$ that characterise the polynomial relations amongst the nonzero entries of the ith Jordan block. We note that in this example each set $S_{2,i}$ is empty because each of the four Jordan blocks has size at most 2. Similarly, the set of polynomial relations S_4 is empty.

Finally, in this example the only non-empty set amongst the $S_{3,i,j}$'s is $S_{3,3,4}$, as the $S_{3,i,j}$ relations require Jordan blocks J_i and J_j of size at least 2. The single polynomial relation we infer is

$$p_3(\boldsymbol{x}) = x_3 x_6 - x_4 x_5.$$

From Claim 3.2, we have $I_R = \langle p_{11}, p_{12}, p_{13}, p_{14}, p_3 \rangle$, or

$$R \cdot \overline{\{J^n \boldsymbol{\beta} : n \in \mathbb{N}\}} = V(I_R).$$

In order to compute the polynomials that define the algebraic set $\overline{O} = \overline{\{M^n \boldsymbol{\alpha} : n \in \mathbb{N}\}}$, we need to apply the transformation $PU^{-1}R^{-1}$ to each of the generators of I_R. For example, we obtain

$$p'_{12}(\boldsymbol{x}) \coloneqq p_{12}(RUP^{-1}\boldsymbol{x}) = \frac{1}{10}(x_1 - ix_2 - x_3 + ix_4 + x_5 - ix_6)(x_1 + ix_2 - x_3 - ix_4 + x_5 + ix_6) - 1.$$

The polynomials $p'_{11}, p'_{13}, p'_{14},$ and p'_3 are obtained similarly. Thus the algebraic set $\overline{O}$ is characterised by the variety $V(\langle p'_{11}, p'_{12}, p'_{13}, p'_{14}, p'_3 \rangle)$ and we have completed the task of invariant generation, as desired.   ◄

Recall that the degree of the splitting field associated with the eigenvalues of a rational matrix may exhibit exponential growth in the dimension of the matrix. In Example 4.3, we consider a loop whose linear update is given by the companion matrix of $(x^2 - 2)(x^2 - 3)(x^2 - 5)$. This matrix has spectrum $\{\pm\sqrt{2}, \pm\sqrt{3}, \pm\sqrt{5}\}$, and the splitting field of its eigenvalues has degree 2^3 over $\mathbb{Q}$. Extending this example, we can construct a matrix with spectrum $\{\pm\sqrt{p_1}, \ldots, \pm\sqrt{p_k}\}$ where p_i is the ith prime number. The degree of the splitting field $\mathbb{Q}(\sqrt{p_1}, \ldots, \sqrt{p_k})$ over $\mathbb{Q}$ is 2^k, while the dimension of the matrix is $2k$.

Example 4.3. We apply our invariant generation procedure for the following loop.

Input: $(x_1, x_2, x_3, x_4, x_5, x_6) \leftarrow$ $\boxed{(6,0,\text{-}62/15,0,2/3,0)}$
    while true **do**

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} \leftarrow \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 30 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -31 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 10 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix}$$

end while

Loop 3. Loop defined by the tuple $\langle M, \boldsymbol{\alpha} \rangle$

A Jordan decomposition (P, J) of the update matrix M is given by the matrices

$$P = \frac{1}{30} \begin{pmatrix} 30 & 30 & 30 & 30 & 30 & 30 \\ 15\sqrt{2} & -15\sqrt{2} & 10\sqrt{3} & -10\sqrt{3} & 6\sqrt{5} & -6\sqrt{5} \\ -16 & -16 & -21 & -21 & -25 & -25 \\ -8\sqrt{2} & 8\sqrt{2} & -7\sqrt{3} & 7\sqrt{3} & -5\sqrt{5} & 5\sqrt{5} \\ 2 & 2 & 3 & 3 & 5 & 5 \\ \sqrt{2} & -\sqrt{2} & \sqrt{3} & -\sqrt{3} & \sqrt{5} & -\sqrt{5} \end{pmatrix}$$

and

$$J = \operatorname{diag}(\sqrt{2}, -\sqrt{2}, \sqrt{3}, -\sqrt{3}, \sqrt{5}, -\sqrt{5}).$$

Since $P^{-1}\boldsymbol{\alpha} = (1, 1, 1, 1, 1, 1)^\top$, the tuple (P, J) is a convenient form for the pair $\langle M, \boldsymbol{\alpha} \rangle$. We now compute the polynomials that define the orbit closure $\overline{O}$. Since each eigenvalue of M is simple, the convenient form immediately leads us to

$$\overline{O} = P \cdot \overline{\left\{ \left((\sqrt{2})^n, (-\sqrt{2})^n, (\sqrt{3})^n, (-\sqrt{3})^n, (\sqrt{5})^n, (-\sqrt{5})^n \right)^\top : n \in \mathbb{N} \right\}}.$$

Then the ideal defining $P^{-1}\overline{O}$ is

$$I = \langle x_1^2 - x_2^2, x_3^2 - x_4^2, x_5^2 - x_6^2 \rangle,$$

and after applying matrix P, we obtain

$$\overline{O} = V\left(\left\langle \begin{matrix} x_3 x_4 + x_2 x_5 + 10 x_4 x_5 + x_1 x_6 + 10 x_3 x_6 + 69 x_5 x_6, \\ x_2 x_3 + x_1 x_4 - 31 x_4 x_5 - 31 x_3 x_6 - 280 x_5 x_6, \\ x_1 x_2 + 30 x_4 x_5 + 30 x_3 x_6 + 300 x_5 x_6 \end{matrix} \right\rangle \right).$$

◀

The complexity of our algorithm for computing the orbit closure of simple linear loops primarily depends on finding a basis for the lattice of multiplicity relations between the eigenvalues of matrices. In particular, as argued in the proof of Theorem 1.1, all other steps in our algorithm run in polynomial time. Ge's algorithm [33] computes such a basis in polynomial time in the degree of the splitting field of the input algebraic numbers (i.e., the update matrix eigenvalues). For rational matrices, while the degree of the eigenvalues is bounded by the matrix dimension, the degree of the splitting field can grow exponentially with the dimension. We observed this in the earlier extension of Example 4.3 to a family of $2k$-dimensional matrices, where the degree of the splitting field becomes 2^k. Therefore, Ge's algorithm, in its current form, does not allow us to reduce the complexity from **PSPACE** to polynomial time.

5 Invariant Verification

Recall that the invariant verification problem checks whether the algebraic set defined by a given set of input polynomials is an invariant for an input loop, though it may not be an inductive invariant. The study and motivation behind such non-inductive invariants have been explored previously [11, 42, 51, 54, 61, 67].

As discussed in Section 2, a standard backward algorithm, used in similar settings in [11, 13, 47, 48], provides a conceptually simple procedure for invariant verification. Let $\langle M, \boldsymbol{\alpha} \rangle$ be a simple linear loop. Let $S \subseteq \mathbb{Q}[\boldsymbol{x}]$ be a set of polynomials, written in the dense representation, with the description size s. Define the sequence of nested ideals $(I_i)_{i \in \mathbb{N}}$ by $I_0 = \langle S \rangle$ and $I_i = \langle P(M^j \boldsymbol{x}) : P \in S, j \leq i \rangle$, as defined in (2). Since $\mathbb{Q}[\boldsymbol{x}]$ is Noetherian, this ascending sequence $I_0 \subseteq I_1 \subseteq I_2 \subseteq \cdots$ of ideals stabilises: there exists k such that $I_k = I_{k+j}$ for all $j \in \mathbb{N}$. Denote by I_∞ the stabilising value of the sequence. The algorithm tests whether $\boldsymbol{\alpha} \in V(I_\infty)$. If yes, then $V(S)$ is an algebraic invariant for $\langle M, \boldsymbol{\alpha} \rangle$; otherwise, it is not.

Since $M\boldsymbol{x}$ is a linear transformation, the degree of $P(M^i \boldsymbol{x})$ is at most the degree of $P(\boldsymbol{x})$. From this observation we prove that if $\boldsymbol{\alpha} \notin V(I_\infty)$ holds then there exists $k = O(2^s)$ such that $\boldsymbol{\alpha} \notin V(I_k)$. Below, we use this bound to obtain a **coNP** upper bound for invariant verification with the input ideal given in dense representation.

PROPOSITION 2.1. *Given $S \subseteq \mathbb{Q}[\boldsymbol{x}]$ in dense representation and a loop with orbit O, verifying whether $O \subseteq V(S)$ is* **coNP**-*complete.*

PROOF. Recall that s denotes the description size of S, and recall the chain of nested ideals $(I_i)_{i \in \mathbb{N}}$, defined above (and in (2)). To show the **coNP** membership, it suffices to provide a polynomial-time verifiable certificate showing that $\boldsymbol{\alpha} \notin V(I_\infty)$ for negative instances of the problem. We first prove that

CLAIM 5.1. *Suppose that $\boldsymbol{\alpha} \notin V(I_\infty)$. There exists $k = O(2^s)$ such that $\boldsymbol{\alpha} \notin V(I_k)$.*

PROOF. By definition, the maximal degree D of polynomials in S and the number d of variables are both bounded by s. For all $n \in \mathbb{N}$ and all polynomials $P \in S$, the degree of $P(M^n \boldsymbol{x})$ is at most D, meaning that the ideal I_n is generated by polynomials of degree at most D.

Let $N \in \mathbb{N}$ be such that $I_N = I_{N+1}$. This implies that $Q(M^{N+1} \boldsymbol{x}) \in \langle P(M^j \boldsymbol{x}) : P \in S, j \leq N \rangle$ for all $Q \in S$. But then $Q(M^{N+2} \boldsymbol{x}) \in \{P(M^j \boldsymbol{x}) : P \in S, j \leq N+1\} = I_{N+1} = I_N$ for all $Q \in S$. Hence, using an inductive argument we can show $I_{N+j} = I_N$ for all $j \in \mathbb{N}$. Let k be the smallest integer such that $I_k = I_{k+1}$, which implies $I_k = I_\infty$. Since $I_0 \subsetneq I_1 \subsetneq \cdots \subsetneq I_k$, at each step i we introduce at least one generator of I_i that can not be expressed as a linear combination of the generators of I_{i-1}. The generators of I_i lie in the vector space of polynomials of degree at most D with dimension $\binom{D+d}{d}$. Therefore, $k \leq \binom{D+d}{d}$ which completes the proof of the claim.   □

This claim implies that there exists $P \in S$ such that $P(M^k \boldsymbol{x})(\boldsymbol{\alpha}) \neq 0$. A **coNP** algorithm guesses $P(\boldsymbol{x}) \in S$, an index $k = O(2^s)$ and a prime p with bitsize s such that $P(M^k \boldsymbol{x})(\boldsymbol{\alpha}) \neq 0$. Taking the binary representation of k into account and, by standard doubling techniques, the algorithm constructs a small circuit for $P(M^k \boldsymbol{x})$ and checks whether $P(M^k \boldsymbol{x}) \not\equiv 0 \pmod{p}$ in polynomial time. We borrow the correctness of the latter test from the well-known fingerprinting procedure for the ACIT problem [3].

The proof of **coNP**-hardness is by a reduction from 3SAT to the complement of the invariant verification problem. Given a 3SAT formula Φ, we construct a loop $\langle M, \boldsymbol{\alpha} \rangle$ with orbit O and a polynomial $Q(\boldsymbol{x})$ such that Φ is satisfiable if and only if $O \nsubseteq V(\langle Q \rangle)$.

Let $\Phi = \bigwedge_{i=1}^m C_m$ be in CNF over variables $\{y_1, \ldots, y_k\}$. Let $p_1 < \cdots < p_k$ be the first k primes. Define $D(i) := 1 + \sum_{j=1}^{i-1} p_j$ for $i \in \{1, \ldots, k\}$ and $d := \sum_{j=1}^{k} p_j$. Construct $M \in \{0, 1\}^{d \times d}$ and $\boldsymbol{\alpha} \in$

$\{0, 1\}^d$ as follows: the entry α_ℓ of $\boldsymbol{\alpha}$ is 1 if, and only if, $\ell = D(i)$ for some $i \in \{1, \dots, k\}$. The matrix M is a block-diagonal matrix $M_1 \oplus \cdots \oplus M_k$, where M_j is a permutation matrix of size p_j. Intuitively speaking, for any $n \in \mathbb{N}$ the vector $M^n \boldsymbol{\alpha}$ splits into k blocks of prime size, and exactly one entry in each block is set to 1. By the Chinese remainder theorem, for every $\boldsymbol{\beta} \in \{0, 1\}^d$ with exactly one 1 entry in each block, there exists $n \in \mathbb{N}$ such that $\boldsymbol{\beta} = M^n \boldsymbol{\alpha}$.

We construct the polynomial Q over the vector $\boldsymbol{x} = (x_1, \cdots, x_d)$ of variables. For each $1 \le i \le k$, we interpret the variables $x_{D(i)}$ and $x_{D(i)+1}$ as literals y_i and $\neg y_i$. The other variables are called *non-literal*. Define $Q \in \mathbb{Q}[\boldsymbol{x}]$ as follows

$$Q(\boldsymbol{x}) := \left(\prod_{x_i \text{ non-literal}} (1 - x_i) \right) \cdot \prod_{i=1}^{m} Q_i(\boldsymbol{x}),$$

where $Q_i := \sum_{j=1}^{k} t_{ij}^2$ with

$$t_{ij} = \begin{cases} x_{D(j)+1} & \text{if } y_j \text{ appears in } C_i, \\ x_{D(j)+2} & \text{if } \neg y_j \text{ appears in } C_i, \\ 0 & \text{otherwise.} \end{cases}$$

The correctness of reduction follows from two simple observations: $Q(\boldsymbol{x})$ vanishes on $\boldsymbol{\beta} \in \{0, 1\}^d$ if $\beta_i = 1$ for some non-literal entry i. Furthermore, provided that all non-literal entries are zero, the point $\boldsymbol{\beta}$ is a zero of $Q(\boldsymbol{x})$ if and only if it corresponds to an unsatisfying assignment of Φ. □

The backward algorithm employed in Proposition 2.1 results in an inefficient **EXPSPACE** upper bound when the input polynomials have sparse representation. Here our route to obtaining a tighter complexity bound is through invariant generation, specifically computing the strongest (inductive) algebraic invariant $\overline{O}$. By Theorem 1.1, we can construct a set $\{I_1, \dots, I_k\}$ of ideals such that $\overline{O} = V(I_1 \cap \cdots \cap I_k)$. The generating set of each ideal I_i has a small cardinality, and comprises of polynomials in $\mathbb{Q}[\boldsymbol{x}]$ written in sparse representation with size polynomial in the input loop description. We perform several radical membership tests to determine the relation of $\overline{O}$ with $V(S)$.

PROPOSITION 2.2. *Given $S \subseteq \mathbb{Q}[\boldsymbol{x}]$ and a loop with the orbit O, verifying whether $\overline{O} \subseteq V(S)$ holds is in* **PSPACE**, *and is* **coNP**-*hard. Similarly, the test $\overline{O} = V(S)$ can be performed in* **PSPACE**. *For all fixed $d \in \mathbb{N}$, both problems for d-dimensional loops are decidable in polynomial time.*

PROOF. Let $\langle M, \boldsymbol{\alpha} \rangle$ be the input loop with orbit O. With Theorem 1.1 at our disposal, we construct the list of generators for the ideals $\mathcal{I}_0, \dots, \mathcal{I}_{n_0-1} \subseteq \mathbb{Q}[\boldsymbol{x}]$ and $\mathcal{Y} \subseteq \mathbb{Q}[\boldsymbol{x}, y_1, \dots, y_s]$. Our construction is such that, for all $i \in \{0, \dots, n_0 - 1\}$, we have $\{M^i \boldsymbol{\alpha}\} = V(\mathcal{I}_i)$. Moreover, $(\boldsymbol{v}, \lambda_1, \dots, \lambda_s) \in V(\mathcal{Y})$ if and only if $\boldsymbol{v} \in \overline{\{M^n \boldsymbol{\alpha} : n \ge n_0\}}$.

We now demonstrate how to check in **PSPACE** whether

$$V(\mathcal{I}_0 \cap \cdots \cap \mathcal{I}_{n_0-1} \cap \mathcal{Y}) \sim V(S)$$

holds for $\sim \in \{=, \subseteq\}$. Since $\mathcal{Y}$ is defined with extra variables y_i's, we add the relations between these extra variables to S. (More precisely, for each λ_i, $1 \le i \le s$, we add the set of polynomials defined in (13), together with $\tau(S_1)$, $\tau(PJP^{-1}) - M$ and $\tau(UP^{-1})\boldsymbol{\alpha} - \boldsymbol{\beta}$ used in definition of $\mathcal{Y}$ from $\mathcal{J}$.) We first explain how to check $\overline{O} \subseteq V(S)$ in **PSPACE**. Towards this, we first check if the rational points $M^i \boldsymbol{\alpha}$ with $i \in \{0, \dots, n_0 - 1\}$ lie in $V(S)$, this reduces to the ACIT and can be tested in randomised polynomial time [3]. It remains to check whether $V(\mathcal{J}) \subseteq V(S)$; we verify this through testing whether $P \in \sqrt{\mathcal{J}}$ for each polynomial $P \in S$, which reduces to radical membership testing. As explained in Section 2, the latter task is in **AM** under GRH and in **PSPACE** unconditionally.

To conclude, we explain how to test $V(S) \subseteq \overline{O}$ in PSPACE. We test whether every polynomial $P \in \mathcal{I}_0 \cap \cdots \cap \mathcal{I}_{n_0-1} \cap \mathcal{J}$ is a member of $\sqrt{S}$. This test could be algorithmically expensive due to the intersection, as a generating set for the intersection of these ideals can face an exponential blow-up in size. We instead test the complement of this question, by guessing one generator from each $\mathcal{I}_i$ and one from $\mathcal{J}$ and checking whether the product is not a member of $\sqrt{S}$. Here again we rely on the AM protocol of radical membership testing.

The hardness follows from the construction given in Proposition 2.1.　　　　□

6　Bit-bounded Synthesis

The results in this section are motivated by the HTP-hardness results in Lemma 2.3. Herein we consider complexity bounds for variants of the loop synthesis problem where one bounds the bitsizes of the loop components (Propositions 2.4 and 2.5). In Proposition 2.5, we obtain tighter bounds by placing an additional dimension specification on the loop.

PROPOSITION 2.4. *The strong and weak bit-bounded synthesis problems over $\{\mathbb{Q}, \mathbb{Z}\}$ lie in* PSPACE. *The weak variant is* NP-*hard; and the strong variant is* NP-*hard under randomised reductions.*

PROOF. The PSPACE bound follows by guessing M and $\boldsymbol{\alpha}$ with entries respecting the required bit bounds, and using the invariant verification subroutine in Proposition 2.2 applied to $\langle M, \boldsymbol{\alpha} \rangle$ and the input ideal. Thus all that remains is to prove the claimed lower bounds.

Our lower bounds are obtained by reductions from 3SAT, and Unique 3SAT, following the folklore encoding of 3SAT in HN [53]. The following encoding is the main building block for both reductions. Let $\Phi := \bigwedge_{i=1}^{m} C_m$ be in CNF over variables $\{y_1, \ldots, y_d\}$. From Φ we construct a set of polynomials $S \subseteq \mathbb{Q}[\boldsymbol{x}]$ with vector $\boldsymbol{x} = (x_1, \ldots, x_{d+2})$ of variables. We initialise S with polynomial $x_{d+1} - x_{d+2}$. For each boolean variable y_i, with $i \in \{1, \ldots, d\}$, we add the polynomial $x_i(1 - x_i)$ to S. For each clause C_j, with $j \in \{1, \ldots, m\}$, we add $P_i := \prod_{j=1}^{d} t_{ij}$ to S, where

$$t_{ij} = \begin{cases} 1 - x_j & \text{if } y_j \text{ appears in } C_i, \\ x_j & \text{if } \neg y_j \text{ appears in } C_i, \\ 1 & \text{otherwise.} \end{cases} \tag{15}$$

Weak bit-bounded synthesis over $\{\mathbb{Q}, \mathbb{Z}\}$ is NP-hard:
The proof is by a reduction from 3SAT. Given an instance Φ of 3SAT, construct the set S of polynomials, as described above.

Assume that Φ is satisfiable. Given a satisfying assignment, define $\boldsymbol{\alpha}$ such that $\alpha_i = 1$ if and only if y_i is true in the assignment, and set $\alpha_{d+1} = \alpha_{d+2} = 1$. Clearly, the infinite orbit of the point $\boldsymbol{\alpha}$ under the matrix $M := \operatorname{diag}(1, \ldots, 1, 2, 2)$ lies in $V(\langle S \rangle)$. Conversely, if Φ is unsatisfiable, the set $V(\langle S \rangle)$ is empty and, a fortiori, no loop exists for $\langle S \rangle$.

We note in passing that that the entries of such M and $\boldsymbol{\alpha}$ have constant bitsize.

Strong bit-bounded synthesis is NP-hard under randomised reductions:
Recall that Unique 3SAT is NP-hard under randomised polynomial-time reductions [71]. It is a promised version of 3SAT, where the input formula is *promised* to have *at most one* satisfying assignment.

Given an instance Φ of Unique 3SAT, construct again the set S of polynomials, as described above. By the promise on Φ, the projection of $V(\langle S \rangle)$ into the first d coordinates is either empty or a singleton. The proof is immediate from the previous case, and by the observation that if Φ is satisfiable, the projection of $\overline{\{M^n \boldsymbol{\alpha} : n \in \mathbb{N}\}}$ into the last two coordinates is $V(\langle x_{d+1} - x_{d+2} \rangle)$.　　□

We now improve the complexity lower bounds under an additional dimensionality assumption.

PROPOSITION 2.5. *For all fixed $d \in \mathbb{N}$, the strong and weak bit-bounded synthesis problems in dimension d over $\{\mathbb{Q}, \mathbb{Z}\}$ are in* NP. *Moreover, weak bit-bounded synthesis over $\mathbb{Z}$ is* NP-*complete.*

PROOF. The NP bound follows by guessing M and $\boldsymbol{\alpha}$ with entries respecting the required bit bounds, and using the invariant verification subroutine in Proposition 2.2 applied to $\langle M, \boldsymbol{\alpha} \rangle$ and the input ideal.

The NP lower bound is by reduction from the quadratic Diophantine equations problem, known to be NP-complete [32, 56]: it asks, given natural numbers a, b, c, whether there a solution $(x, y) \in \mathbb{N}^2$ to the equation $ax^2 + by = c$. Given an instance (a, b, c) of the quadratic Diophantine equations problem (a, b, c), we construct polynomials $L, P, Q \in \mathbb{Z}[\boldsymbol{x}]$ with vector $\boldsymbol{x} = (x, y, y_1, y_2, y_3, y_4, z_1, z_2)$ of variables such that there is an integer point in the variety $V(\langle L, P \rangle) \subseteq \overline{\mathbb{Q}}^6$ if and only if the original equation $ax^2 + by = c$ has a solution $(x, y) \in \mathbb{N}^2$. Define

$$L(\boldsymbol{x}) := y - y_1^2 - y_2^2 - y_3^2 - y_4^2,$$
$$P(\boldsymbol{x}) := ax^2 + by - c,$$
$$Q(\boldsymbol{x}) := z_1 - z_2.$$

By Lagrange's four-squares theorem, every positive integer y can be expressed as a sum of four integer squares; thus polynomial L ensures that variable y can attain only non-negative integer values. Suppose the intersection $V(\langle L, P, Q \rangle) \cap \mathbb{Z}^8$ is non-empty. Observe that for $\boldsymbol{\alpha} \in V(\langle L, P, Q \rangle) \cap \mathbb{Z}^8$, the orbit under $M := \mathrm{diag}(1, \ldots, 1, 2, 2)$ is infinite, and thus $\langle M, \boldsymbol{\alpha} \rangle$ is a non-trivial loop. The converse direction is immediate. □

7 Further Discussion

We suggest several directions for further research inspired by our contributions to invariant generation for simple linear loops and loop synthesis presented herein.

Invariant Generation for Affine Programs. A program is considered affine if it exclusively features nondeterministic branching (as opposed to conditional branching) and all its assignments are defined by affine expressions. The invariant generation problem for affine programs is addressed by the algorithm in [39] through the group-closure problem. This problem entails computing a generating set of polynomials for the Zariski closure $\overline{\langle M_1, \ldots, M_k \rangle}$ for a given set $\{M_1, \ldots, M_k\}$ of invertible rational matrices. The tightest complexity bound for solving the group-closure problem is severalfold exponential time [63].

The main result in this paper, Theorem 1.1, presents a PSPACE algorithm for generating the invariants of a simple linear loop (the class corresponding to branch-free loops with a single linear update). To extend our technique to the general case of affine programs, we may first consider the setting with multiple linear updates $M_1, \ldots, M_k$ where the matrices are commutative and invertible. Since the M_i commute, the orbit of the loop is defined by $O = \{M_1^{n_1} \cdots M_k^{n_k} \boldsymbol{\alpha} : n_1, \ldots, n_k \in \mathbb{N}\}$ where $\boldsymbol{\alpha}$ is the initial vector.

Recall that a matrix $M \in \overline{\mathbb{Q}}^{d \times d}$ is *unipotent* if there exists $n \in \mathbb{N}$ such that $(M - \mathrm{Id}_d)^n = 0_d$ (here Id_d and 0_d are the $d \times d$ identity and zero matrices, respectively) and M is *semisimple* if it is diagonalisable over $\overline{\mathbb{Q}}$. Define $G := \overline{\langle M_1, \ldots, M_k \rangle}$ so that $\overline{O} = \overline{G\boldsymbol{\alpha}}$. It is known that the subset of semisimple matrices in G, denoted by G_s, forms an algebraic subgroup; likewise the set of unipotent matrices in G, denoted by G_u, forms an algebraic subgroup. By the Jordan–Chevalley decomposition, we have $\overline{O} = \overline{G_u G_s \boldsymbol{\alpha}}$.

In the case that $G = G_s$ we have the following.

LEMMA 7.1. *Let G be a semisimple commutative group. A set of polynomials defining $\overline{G\boldsymbol{\alpha}}$ is computable in* PSPACE.

PROOF. Since G is a semisimple commutative group, there exists $P \in \mathrm{GL}_d(\overline{\mathbb{Q}})$ such that for all $i \in \{1, \ldots, k\}$, the matrix $D_i := P^{-1} M_i P$ is diagonal. Following Lemma 3.1, we can choose P in such a way that $P^{-1}\boldsymbol{\alpha} \in \{0, 1\}^d$. Writing $\overline{O} = P \langle D_1, \ldots, D_k \rangle P^{-1}\boldsymbol{\alpha}$, we observe that $\overline{\langle D_1, \ldots, D_k \rangle P^{-1}\boldsymbol{\alpha}}$ is the closure of a group of diagonal matrices of dimension at most d. To obtain a PSPACE procedure it suffices to closely follow our construction in Theorem 1.1. The sole difference from that construction lies in the set S_1 of polynomial equations, which is now defined by the intersection of lattices L_i of multiplicity relations between the entries of $D_i P^{-1}\boldsymbol{\alpha}$ [16, Chapter 3]. □

A natural direction for future research involves extending our PSPACE procedure to apply to commutative matrices more broadly. This requires a better understanding of the polynomial map

$$G_u \times \overline{G_s \boldsymbol{\alpha}} \to \overline{G \boldsymbol{\alpha}},$$

$$(g, \boldsymbol{v}) \mapsto g\boldsymbol{v}.$$

Subsequently, an ambitious objective is to develop a procedure with improved complexity bounds for the invariant generation problem in affine programs.

Loop Synthesis. Our results for bit-bounded synthesis over the rationals (Propositions 2.4 and 2.5) demonstrate an inherent source of hardness (Hilbert's Tenth problem). A first direction for future research might consider circumventing such obstacles by focusing on classes of ideals with an abundance of rational solutions. A small initial step in this direction, the synthesis of loops for pure-difference binomials, was shown in [50]. An ultimate goal is the synthesis of loops for the larger class of binomial ideals.

Acknowledgments

References

[1] Rida Ait El Manssour, George Kenison, Mahsa Shirmohammadi, and Anton Varonka. 2024. Simple linear loops: algebraic invariants and applications. (2024). arXiv: 2407.09154v2 [cs.CC].

[2] Rida Ait El Manssour, George Kenison, Mahsa Shirmohammadi, and James Worrell. 2024. Determination problems for orbit closures and matrix groups. (2024). arXiv: 2407.04626 [cs.CC].

[3] Eric Allender, Peter Bürgisser, Johan Kjeldgaard-Pedersen, and Peter Bro Miltersen. 2009. On the complexity of numerical analysis. *SIAM Journal on Computing*, 38, 5, 1987–2006.

[4] Rajeev Alur, Rishabh Singh, Dana Fisman, and Armando Solar-Lezama. 2018. Search-based program synthesis. *Communications of the ACM*, 61, 12, 84–93.

[5] Daneshvar Amrollahi, Ezio Bartocci, George Kenison, Laura Kovács, Marcel Moosbrugger, and Miroslav Stankovič. 2024. (Un)Solvable loop analysis. *Formal Methods in System Design*. https://doi.org/10.1007/s10703-024-00455-0.

[6] Daneshvar Amrollahi, Ezio Bartocci, George Kenison, Laura Kovács, Marcel Moosbrugger, and Miroslav Stankovič. 2022. Solving Invariant Generation for Unsolvable Loops. In *Static Analysis*, 19–43. https://doi.org/10.1007/978-3-031-22308-2_3.

[7] Sanjeev Arora and Boaz Barak. 2009. *Computational complexity: a modern approach*. Cambridge University Press.

[8]   László Babai and Shlomo Moran. 1988. Arthur–Merlin games: A randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36, 2, 254–276. https://doi.org/10.1016/0022-0000(88)900 28-1.

[9]   Nikhil Balaji, Klara Nosan, Mahsa Shirmohammadi, and James Worrell. 2022. Identity testing for radical expressions. In *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, 8:1–8:11.

[10]   Nikhil Balaji, Sylvain Perifel, Mahsa Shirmohammadi, and James Worrell. 2021. Cyclotomic identity testing and applications. In *ISSAC '21: International Symposium on Symbolic and Algebraic Computation, Virtual Event, Russia, July 18-23, 2021*, 35–42. https://doi.org/10.1145/3452143.3465530.

[11]   Erdenebayar Bayarmagnai, Fatemeh Mohammadi, and Rémi Prébet. 2024. Algebraic tools for computing polynomial loop invariants. In *Proceedings of the 2024 International Symposium on Symbolic and Algebraic Computation, ISSAC 2024, (To Appear)*. https://doi.org/10.1145/3666000.3669710.

[12]   Robert Benedetto, Patrick Ingram, Rafe Jones, Michelle Manes, Joseph H. Silverman, and Thomas J. Tucker. 2019. Current trends and open problems in arithmetic dynamics. *Bull. Amer. Math. Soc. (N.S.)*, 56, 4, 611–685. https://doi.or g/10.1090/bull/1665.

[13]   Michael Benedikt, Timothy Duff, Aditya Sharad, and James Worrell. 2017. Polynomial automata: zeroness and applications. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 1–12.

[14]   D. Beyer, T. A. Henzinger, R. Majumdar, and A. Rybalchenko. 2007. Invariant synthesis for combined theories. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*. Springer, 378–394.

[15]   Markus Bläser, Christian Ikenmeyer, Vladimir Lysikov, Anurag Pandey, and Frank-Olaf Schreyer. 2021. On the orbit closure containment problem and slice rank of tensors. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, 2565–2584. https://doi.org/10.1137/1.9781611976465 .152.

[16]   Enrico Bombieri and Walter Gubler. 2006. *Heights in Diophantine geometry*. Number 4. Cambridge university press.

[17]   Peter Bürgisser. 2024. Completeness classes in algebraic complexity theory. *arXiv preprint arXiv:2406.06217*.

[18]   Peter Bürgisser, Joseph M Landsberg, Laurent Manivel, and Jerzy Weyman. 2011. An overview of mathematical issues arising in the geometric complexity theory approach to vp≠vnp. *SIAM Journal on Computing*, 40, 4, 1179–1209.

[19]   Jin-Yi Cai. 1994. Computing jordan normal forms exactly for commuting matrices in polynomial time. *International Journal of Foundations of Computer Science*, 05, 03n04, 293–302. https://doi.org/10.1142/S0129054194000165.

[20]   Jin-yi Cai, Richard J. Lipton, and Yechezkel Zalcstein. 2000. The Complexity of the A B C Problem. *SIAM Journal on Computing*, 29, 6, 1878–1888. https://doi.org/10.1137/S0097539794276853.

[21]   Emmanuel J. Candes and Terence Tao. 2010. The power of convex relaxation: near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56, 5, 2053–2080. https://doi.org/10.1109/tit.2010.2044061.

[22]   Henri Cohen. 2013. *A course in computational algebraic number theory*. Vol. 138. Springer Science & Business Media.

[23]   David A. Cox, John B. Little, and Donal O'Shea. 2015. *Ideals, varieties, and algorithms*. (Fourth ed.). An introduction to computational algebraic geometry and commutative algebra. Springer, Cham, xvi+646. https://doi.org/10.1007/978 -3-319-16721-3.

[24]   John Cyphert and Zachary Kincaid. 2024. Solvable polynomial ideals: the ideal reflection for program analysis. *Proc. ACM Program. Lang.*, 8, POPL, Article 25, 29 pages. https://doi.org/10.1145/3632867.

[25]   James H. Davenport and Joos Heintz. 1988. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5, 1, 29–35. https://doi.org/10.1016/S0747-7171(88)80004-X.

[26]   Martin Davis. 1972. On the Number of Solutions of Diophantine Equations. *Proceedings of the American Mathematical Society*, 35, 2, 552–554.

[27]   W. A. de Graaf. 2017. *Computation with Linear Algebraic Groups*. CRC Press.

[28]   H. Derksen, E. Jeandel, and P. Koiran. 2005. Quantum automata and algebraic groups. *J. Symb. Comput.*, 39, 3-4, 357–371.

[29]   Klaus Dräger. 2016. The invariance problem for matrix semigroups. In *Foundations of Software Science and Computation Structures*, 479–492.

[30]   Michael A Forbes and Amir Shpilka. 2013. Explicit noether normalization for simultaneous conjugation via polynomial identity testing. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer, 527–542.

[31]   Francesco Galuppi and Mima Stanojkovski. 2021. Toric varieties from cyclic matrix semigroups. *Rend. Istit. Mat. Univ. Trieste*. https://doi.org/10.13137/2464-8728/33099.

[32]   M. R. Garey and D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. (First Edition ed.). W. H. Freeman.

[33]   Guoqiang Ge. 1993. *Algorithms Related to Multiplicative Representations of Algebraic Numbers*. Ph.D. Dissertation. U.C. Berkeley.

[34] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35, 12, 61–70. https://doi.org/10.1145/138859.138867.

[35] Daniel R. Grayson and Michael E. Stillman. 2002. Macaulay2, a software system for research in algebraic geometry. Available at http://www2.macaulay2.com. (2002).

[36] Sumit Gulwani, Oleksandr Polozov, and Rishab Singh. 2017. Program synthesis. *Foundations and Trends® in Programming Languages*, 4, 1-2, 1–119.

[37] S. Hitarth, George Kenison, Laura Kovács, and Anton Varonka. 2024. Linear Loop Synthesis for Quadratic Invariants. In *41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024)*. Vol. 289, 41:1–41:18. https://doi.org/10.4230/LIPIcs.STACS.2024.41.

[38] Mehran Hosseini, Joël Ouaknine, and James Worrell. 2019. Termination of linear loops over the integers. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*. Vol. 132, 118:1–118:13.

[39] Ehud Hrushovski, Joël Ouaknine, Amaury Pouly, and James Worrell. 2023. On Strongest Algebraic Program Invariants. *J. ACM*, 70, 5, Article 29, 22 pages. https://doi.org/10.1145/3614319.

[40] Ehud Hrushovski, Joël Ouaknine, Amaury Pouly, and James Worrell. 2018. Polynomial invariants for affine programs. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, 530–539.

[41] Andreas Humenberger, Daneshvar Amrollahi, Nikolaj Bjørner, and Laura Kovács. 2022. Algebra-Based Reasoning for Loop Synthesis. *Form. Asp. Comput.*, 34, 1, Article 4, 31 pages. https://doi.org/10.1145/3527458.

[42] Andreas Humenberger, Nikolaj S. Bjørner, and Laura Kovács. 2020. Algebra-Based Loop Synthesis. In *Integrated Formal Methods - 16th International Conference, IFM 2020, Lugano, Switzerland, November 16-20, 2020, Proceedings*. Vol. 12546, 440–459. https://doi.org/10.1007/978-3-030-63461-2_24.

[43] Andreas Humenberger, Maximilian Jaroschek, and Laura Kovács. 2017. Automated generation of non-linear loop invariants utilizing hypergeometric sequences. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, 221–228. https://doi.org/10.1145/3087604.3087623.

[44] Andreas Humenberger, Maximilian Jaroschek, and Laura Kovács. 2018. Invariant generation for multi-path loops with polynomial assignments. In *Verification, Model Checking, and Abstract Interpretation*, 226–246.

[45] Charles R Johnson. 1990. Matrix completion problems: a survey. In *Matrix theory and applications*. Vol. 40, 171–198.

[46] Michael Karr. 1976. Affine relationships among variables of a program. *Acta Informatica*, 6, 133–151.

[47] Manuel Kauers. 2007. An algorithm for deciding zero equivalence of nested polynomially recurrent sequences. *ACM Trans. Algorithms*, 3, 2, Art. 18, 14. https://doi.org/10.1145/1240233.1240241.

[48] Manuel Kauers. 2008. Solving difference equations whose coefficients are not transcendental. *Theoret. Comput. Sci.*, 401, 1-3, 217–227. https://doi.org/10.1016/j.tcs.2008.05.001.

[49] Manuel Kauers and Burkhard Zimmermann. 2008. Computing the algebraic relations of C-finite sequences and multisequences. *Journal of Symbolic Computation*, 43, 11, 787–803. https://doi.org/10.1016/j.jsc.2008.03.002.

[50] George Kenison, Laura Kovács, and Anton Varonka. 2023. From Polynomial Invariants to Linear Loops. In *Proceedings of the 2023 International Symposium on Symbolic and Algebraic Computation, ISSAC 2023, Tromsø, Norway, July 24-27, 2023*, 398–406. https://doi.org/10.1145/3597066.3597109.

[51] Zachary Kincaid, John Cyphert, Jason Breck, and Thomas Reps. 2017. Non-linear reasoning for invariant synthesis. *Proc. ACM Program. Lang.*, 2, POPL, Article 54, 33 pages. https://doi.org/10.1145/3158142.

[52] P. Koiran. 1997. Randomized and deterministic algorithms for the dimension of algebraic varieties. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, 36–45.

[53] Pascal Koiran. 1996. Hilbert's Nullstellensatz Is in the Polynomial Hierarchy. *J. Complex.*, 12, 4, 273–286. long version, DIMACS report 96-27. https://doi.org/10.1006/jcom.1996.0019.

[54] Laura Kovács. 2008. Reasoning algebraically about p-solvable loops. In *Tools and Algorithms for the Construction and Analysis of Systems*, 249–264.

[55] Rupak Majumdar, Joël Ouaknine, Amaury Pouly, and James Worrell. 2020. Algebraic Invariants for Linear Hybrid Automata. In *31st International Conference on Concurrency Theory (CONCUR 2020)*. Vol. 171, 32:1–32:17. https://doi.org/10.4230/LIPIcs.CONCUR.2020.32.

[56] Kenneth L. Manders and Leonard Adleman. 1978. NP-Complete decision problems for binary quadratics. *Journal of Computer and System Sciences*, 16, 2, 168–184. https://doi.org/10.1016/0022-0000(78)90044-2.

[57] D. W. Masser. 1988. Linear relations on algebraic groups. In *New Advances in Transcendence Theory*, 248–262. https://doi.org/10.1017/CBO9780511897184.016.

[58] Yuri Matiyasevich. 1970. Enumerable sets are Diophantine. *Soviet Math. Dokl.*, 11, 354–358.

[59] B. Mishra. 2012. *Algorithmic Algebra*. Springer New York.

[60] Markus Müller-Olm and Helmut Seidl. 2004. A note on Karr's algorithm. In *International Colloquium on Automata, Languages, and Programming*. Springer, 1016–1028.

[61] Markus Müller-Olm and Helmut Seidl. 2004. Computing polynomial program invariants. *Inf. Process. Lett.*, 91, 5, 233–244. https://doi.org/10.1016/J.IPL.2004.05.004.

[62] Julian Müllner, Marcel Moosbrugger, and Laura Kovács. 2024. Strong invariants are hard: on the hardness of strongest polynomial invariants for (probabilistic) programs. *Proc. ACM Program. Lang.*, 8, POPL, Article 30, 29 pages. https://doi.org/10.1145/3632872.

[63] K. Nosan, A. Pouly, S. Schmitz, M. Shirmohammadi, and J. Worrell. 2022. On the Computation of the Zariski Closure of Finitely Generated Groups of Matrices. In *Proceedings of the 2022 International Symposium on Symbolic and Algebraic Computation*, 129–138.

[64] Joël Ouaknine and James Worrell. 2015. On linear recurrence sequences and loop termination. *ACM SIGLOG News*, 2, 2, 4–13. https://doi.org/10.1145/2766189.2766191.

[65] Joël Ouaknine and James Worrell. 2014. Positivity problems for low-order linear recurrence sequences. In *Proceedings of the 2014 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 366–379. https://doi.org/10.1137/1.9781611973402.27.

[66] Bjorn Poonen. 2008. Undecidability in number theory. *Notices of the AMS*, 55, 3.

[67] Enric Rodríguez-Carbonell and Deepak Kapur. 2004. Automatic Generation of Polynomial Loop Invariants: Algebraic Foundations. In *Proc. of ISSAC*, 266–273.

[68] Alexandra Shlapentokh. 2011. Defining integers. *The Bulletin of Symbolic Logic*, 17, 2, 230–251.

[69] Joseph H. Silverman. 2007. *The arithmetic of dynamical systems*. Vol. 241. Springer, New York, x+511. https://doi.org/10.1007/978-0-387-69904-2.

[70] Saurabh Srivastava, Sumit Gulwani, and Jeffrey S Foster. 2013. Template-based program verification and program synthesis. *International Journal on Software Tools for Technology Transfer*, 15, 497–518.

[71] L. G. Valiant and V. V. Vazirani. 1986. NP is as Easy as Detecting Unique Solutions. *Theor. Comput. Sci.*, 47, 1, 85–93.