2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

Complexity of the LTI system trajectory boundedness problem

Guillaume O. Berger and Raphaël M. Jungers

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9683762

***Abstract*— We study the algorithmic complexity of the problem of deciding whether a Linear Time Invariant dynamical system with rational coefficients has bounded trajectories. Despite its ubiquitous and elementary nature in Systems and Control, it turns out that this question is quite intricate, and, to the best of our knowledge, unsolved in the literature. We show that classical tools, such as Gaussian Elimination, the Routh–Hurwitz Criterion, and the Euclidean Algorithm for GCD of polynomials indeed allow for an algorithm that is polynomial in the bit size of the instance. However, all these tools have to be implemented with care, and in a non-standard way, which relies on an advanced analysis.**

I. Introduction

This paper deals with the computational problem of deciding whether a Linear Time Invariant (LTI) dynamical system with rational coefficients has bounded trajectories; see Problems 1 and 2 in Section II. We show that this problem can be solved in polynomial time with respect to the bit size of the coefficients. We are interested in the *exact complexity*, also called "bit complexity" or "complexity in the Turing model", which accounts for the fact that arithmetic operations ($+$, $-$, $\times$, $\div$) on integers and rational numbers take a time proportional to the bit size of the operands.

Rational matrices appear in many applications, including combinatorics, computer science and information theory; for instance, the number of paths of length r in a graph (involved for instance in the computation of its entropy [13]) grows at most as ρ^r ($\rho \geq 0$) if and only if the adjacency matrix of the graph divided by ρ has bounded powers. However, despite its ubiquitous and paradigmatic nature for many applications, it seems that the question of the complexity of the problem of deciding whether the trajectories of a LTI system with rational coefficients are bounded is unsolved in the literature. The aim of this paper is to fill this gap by providing a proof of its polynomial complexity.

The question of deciding *asymptotic stability* (rather than boundedness of the trajectories) of LTI systems has received a lot of attention in the literature [11], [12], [15]. Algorithms with polynomial bit complexity have been proposed to address this question for systems with rational coefficients. This includes algorithms based on the *Routh–Hurwitz stability criterion* [14] where care is taken to avoid exponential blow-up of the bit size of the intermediate coefficients; or algorithms based on the resolution of the *Lyapunov equation* [2], again taking care to avoid exponential blow-up of the bit size of the intermediate steps of the resolution.

GB is a FRIA (F.R.S.–FNRS) fellow. RJ is a FNRS honorary Research Associate. This project has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program under grant agreement No. 864017 – L2C. RJ is also supported by the Walloon Region, the Innoviris Foundation, and the FNRS (Chist-Era Druid-net). Both are with ICTEAM institute, UCLouvain, Belgium. Emails: {guillaume.berger,raphael.jungers}@uclouvain.be

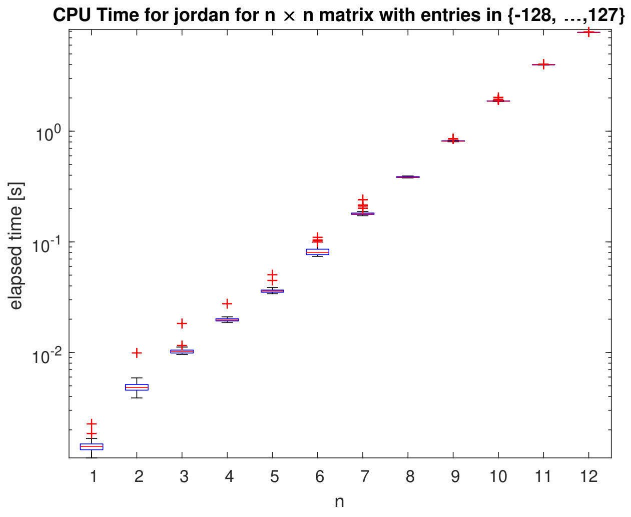

Fig. 1. A naive way to decide the boundedness of the trajectories of a LTI system described by a rational matrix would be to use the Matlab function `jordan` and look at the eigenvalues and size of the associated Jordan blocks. However, this method will not be efficient for two reasons. Firstly, the complexity of the Matlab function `jordan` applied on symbolic integer matrices seems *super-linear*. In the above plot, we have generated, for each $n \in \{1, \ldots, 12\}$, 50 random $n \times n$ matrices with 8-bit integer entries (the bit size of the input is thus $8n^2$) and we measured the time to compute the Jordan form (represented by the boxplots). The empirical complexity is clearly super-linear. Secondly, the obtained Jordan form does not contain explicitly the eigenvalues of the matrix, but rather expressions of the form `root('some polynomial')`, so that extra computation is needed to decide the (marginal) stability of each of the Jordan blocks. In this work, we show that a more efficient implementation is possible.

However, these algorithms focus on the asymptotic stability and do not extend straightforwardly for the problem of deciding boundedness of the trajectories. For instance, the Routh–Hurwitz stability criterion applied on the characteristic polynomial of a matrix allows to decide whether all eigenvalues of the matrix have negative real part. For the problem of boundedness of the trajectories, the analysis is more difficult because eigenvalues with nonnegative real parts are also allowed[1], provided they are on the imaginary axis and correspond to Jordan blocks of size one in the Jordan normal form of the system matrix. Extensions of the Routh–Hurwitz criterion to compute the *number* of roots with negative real part of a given polynomial have also been proposed in the literature (see, e.g., [9, §15], [4], [5]), but the analysis of the bit complexity of these algorithms remains

[1]This holds for the continuous-time case, but a similar result holds for the discrete-time case. Both cases are studied in this paper.

elusive. The situation is similar for the approach based on the Lyapunov equation [2]. More precisely, while for the study of the asymptotic stability, a solution of the Lyapunov matrix inequality can be computed by arbitrarily fixing the right-hand side term to $-I$, this trick cannot be used for the analysis of the boundednes of the trajectories, since the RHS term is not guaranteed to be negative definite. One has thus to solve a matrix *inequality* instead of a matrix *equation*, and there is to the best of the authors' knowledge no clear result available in the literature on the bit complexity of solving LMIs, so that an extension of the Lyapunov method for the problem of trajectory boundedness is not straightforward.

Objectives and methodology. The discussion above nevertheless suggests an algorithmic procedure for our problem, consisting in

1) computing the minimal polynomial of the system matrix, which contains the information on the eigenvalues and on the size of the largest associated Jordan block in the Jordan normal form of the system matrix;
2) using an extension of the Routh–Hurwitz criterion to decide whether all the roots of the minimal polynomial either have negative real part, or are on the imaginary axis and correspond to Jordan blocks of size one.

We provide self-contained proofs that these two steps can be achieved in polynomial time with respect to the bit size of the matrix. In particular, we provide a careful analysis of the extended Routh–Hurwitz criterion, showing that it provides a polynomial bit-complexity algorithm for the second step.

Comparison with the literature. Algorithms with polynomial bit complexity for computing the minimal polynomial of a rational matrix have been proposed in the literature [7], [8].[2] While being aware of these results, we describe here an *elementary* algorithm for this problem, based on the defining property of the minimal polynomial (see Subsection IV-A). The proposed algorithm is likely to be less efficient than those available in the literature, but its elementary nature allows us to provide a simple self-contained proof of its polynomial bit complexity.

As for the second step, several extensions of the Routh–Hurwitz criterion have been proposed in the literature to compute for a given polynomial the number of roots on the imaginary axis and their multiplicity [4], [5], [9]. However, no proofs of the polynomial complexity of these algorithms are provided. In particular, since they are extensions of the classical Routh–Hurwitz criterion, there is no guarantee on the boundedness of the bit size of the intermediate coefficients; see, e.g., [14, p. 321] for a discussion on the "bit size growth factor" for the Routh–Hurwitz criterion. The extended Routh–Hurwitz criterion proposed in this paper draws on these results and combines them with techniques introduced in the context of the classical Routh–Hurwitz criterion to avoid "bit-size blow-up". This results in a sound elementary algorithm to address the second step, and for which we provide a simple self-contained proof of the polynomial bit complexity.

Outline. The paper is organized as follows. The statement of the problems and the main results are presented in Section II. Some preliminary results, namely on the computation of the determinant and the resolution of systems of linear equations, are presented in Section III. Then, in Section IV, we present the proof of the main result for continuous-time systems. Finally, in Section V, we present the proof of the main result for discrete-time systems.

Many of the intermediate results used in our analysis are inspired from classical results, but have been adapted for the needs of this work. Below, we refer to these results as *folk theorems*, meaning that we are referring to a classical result — possibly slightly adapted to our needs, but on which we do not claim any paternity.

Notation. We use a Matlab-like notation for the indexing of submatrices; e.g., $A_{[1:r,:]}$ denotes the submatrix consisting of the r first rows of the matrix A. The degree of a polynomial p is denoted by $\deg p$. We use i to denote the imaginary unit $i = \sqrt{-1}$ *and* as an index $i \in \mathbb{N}$, but the disambiguation should be clear from the context.

II. Problem statement and main results

We start with the definition of bit size for integers, integer matrices and rational matrices.

Definition 1 (Bit size):
- The *bit size* of an integer $a \in \mathbb{Z}$ is $\mathfrak{b}(a) = \lceil \log_2(|a|+1) \rceil + 1$ (= smallest $b \in \mathbb{Z}_{\geq 0}$ such that $-2^{b-1}+1 \leq a \leq 2^{b-1}-1$).
- The *bit size* of an integer matrix $A = (a_{ij})_{i=1,j=1}^{m,n} \in \mathbb{Z}^{m\times n}$ is $\mathfrak{b}(A) = \sum_{i=1,j=1}^{m,n} \mathfrak{b}(a_{ij})$.
- The *bit size* of a rational matrix $A \in \mathbb{Q}^{m\times n}$, described by $A = B/q$ with $B \in \mathbb{Z}^{m\times n}$ and $q \in \mathbb{Z}_{>0}$, is $\mathfrak{b}(A) = \mathfrak{b}(B) + \mathfrak{b}(q)$.

We consider the following decision problem, accounting for the boundedness of the trajectories of continuous-time LTI systems:

Problem 1: Given a rational matrix $A \in \mathbb{Q}^{n\times n}$, decide whether $\sup_{t\in\mathbb{R}_{\geq 0}} \|e^{At}\| < \infty$.

The following theorem states that Problem 1 can be solved in polynomial time with respect to the bit size of the input.

Theorem 1: There is an algorithm that, given any $A \in \mathbb{Q}^{n\times n}$, gives the correct answer to Problem 1, and whose bit complexity is polynomial in $\mathfrak{b}(A)$.

The proof of Theorem 1 is presented in Section IV. Note that a rational matrix $A = B/q$, with $B \in \mathbb{Z}^{n\times n}$ and $q \in \mathbb{R}_{>0}$, is a positive instance of Problem 1 if and only if B is a positive instance of Problem 1. Hence, in Section IV, we limit ourselves to proving Theorem 1 for integer matrices.

The same kind of results can be obtained for the problem of the boundedness of the trajectories of discrete-time LTI systems:

Problem 2: Given a rational matrix $A \in \mathbb{Q}^{n\times n}$, decide whether $\sup_{t\in\mathbb{Z}_{\geq 0}} \|A^t\| < \infty$.

[2]Note that the Matlab function `jordan` would not have helped us for this problem, as the complexity of this function seems to be super-linear in the bit size of the instance; see, e.g., Figure 1.

Theorem 2: There is an algorithm that, given any $A \in \mathbb{Q}^{n\times n}$, gives the correct answer to Problem 2, and whose bit complexity is polynomial in $\mathfrak{b}(A)$.

The proof of Theorem 2 is presented in Section V.

III. Preliminary results

The following result, which follows directly from [1, Eq. (8)], will be instrumental in the following section. Due to space limitation, we only present a sketch of its proof as a corollary of [1, Eq. (8)].

Proposition 3 (from [1]): There is an algorithm that, given any $A \in \mathbb{Z}^{m\times n}$, computes the tuple $(r, \mathcal{R}, \mathcal{C}, \det(A_{[\mathcal{R},\mathcal{C}]}))$ where (i) r is the rank of A, (ii) $\mathcal{R} = \{i_1, \ldots, i_r\} \subseteq \mathbb{N}$ with $1 \le i_1 < i_2 < \ldots < i_r \le m$, (iii) $\mathcal{C} = \{j_1, \ldots, j_r\} \subseteq \mathbb{N}$ with $1 \le j_1 < j_2 < \ldots < j_r \le n$, and (iv) det $A_{[\mathcal{R},\mathcal{C}]} \neq 0$. Moreover, the bit complexity of the algorithm is polynomial in $\mathfrak{b}(A)$.

Proof: See the extended version of this paper [3]. ■

In particular, when $A \in \mathbb{Z}^{n\times n}$, the determinant of A can be obtained from the output $(r, \mathcal{R}, \mathcal{C}, D)$ of the algorithm: if $r < n$, then $\det(A) = 0$, otherwise $\det(A) = D$.

By combining the algorithm of Proposition 3 with the well-known *rule of Cramer* (see, e.g., [10, §0.8.3]), one can obtain a polynomial-time algorithm for the resolution of systems of linear equations with integer coefficients.[3]

Proposition 4: There is an algorithm that, given any $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$, computes integers $x_0 \neq 0$ and $x_1, \ldots, x_n$ such that $x = [x_1, \ldots, x_n]/x_0$ is a solution to $Ax = b$ if the system is feasible, or outputs that the system has no solution (in $\mathbb{R}^n$). Moreover, the bit complexity of the algorithm is polynomial in $\mathfrak{b}(A) + \mathfrak{b}(b)$.

Proof: See the extended version of this paper [3]. ■

IV. Proof of Theorem 1

A. The minimal polynomial

The first step of our algorithm to answer Problem 1 is to compute the minimal polynomial of A. We remind that the *minimal polynomial* of a matrix $A \in \mathbb{R}^{n\times n}$ is defined as the monic real polynomial $p : x \mapsto x^d + c_1 x^{d-1} + \ldots + c_d$ with smallest degree d such that $p(A) = 0$.

The relevance of the minimal polynomial for Problem 1 is explained in Theorem 5 below. First, we introduce the following terminology that will simplify the statement of the theorem.

Definition 2: A (complex or real) polynomial will be said to have the *boundedness property* if each of its roots satisfies one of the following two conditions: (i) has negative real part, or (ii) is on the imaginary axis and is simple.

Theorem 5 (Folk): For any $A \in \mathbb{R}^{n\times n}$, it holds that $\sup_{t\in\mathbb{R}_{\ge 0}} \|e^{At}\| < \infty$ if and only if the minimal polynomial of A has the boundedness property.

Proof: See the extended version of this paper [3]. ■

[3]Let us mention that more efficient algorithms for this problem have been proposed in the literature, such as the well-known *Gaussian Elimination*. However, the latter necessitates more advanced analysis for a careful proof of its polynomial-time nature (see [1, §2]).

Input: $A \in \mathbb{Z}^{n\times n}$.
Output: "YES" if A is a positive instance of Problem 1 and "NO" otherwise.
Algorithm:
▷ **Step 1:** Using Theorem 6, compute integers $e_0 \neq 0$ and $e_1, \ldots, e_d$ such that $x \mapsto x^d + e_1 x^{d-1}/e_0 + \ldots + e_d/e_0$ is the minimal polynomial of A.
Let $p : x \mapsto e_0 x^d + e_1 x^{d-1} + \ldots + e_d$.
▷ **Step 2:** Using Theorem 7, **return** "YES" if p has the boundedness property (see Definition 2) and **return** "NO" otherwise.

Fig. 2. Algorithm for answering Problem 1 for integer matrices (rational matrices can be treated in the same way by considering only the "numerator matrix").

The above theorem allows us to provide an algorithm to answer Problem 1. The algorithm is presented in Figure 2; it consists in two main steps that are described in the following subsections.

B. Step 1: Computation of the minimal polynomial

The definition of the minimal polynomial, combined with the algorithm of Proposition 4, allows for polynomial-time computation of the minimal polynomial of integer matrices.[4]

Theorem 6: There is an algorithm that, given any $A \in \mathbb{Z}^{n\times n}$, computes integers $e_0 \neq 0$ and $e_1, \ldots, e_d$ such that $x \mapsto x^d + e_1 x^{d-1}/e_0 + \ldots + e_d/e_0$ is the minimal polynomial of A. Moreover, the bit complexity of the algorithm is polynomial in $\mathfrak{b}(A)$.

Proof: For each $d \in \{1, \ldots, n\}$, write the matrix equation $A^d + c_1 A^{d-1} + \ldots + c_d I = 0$, with unknowns $c_1, \ldots, c_d \in \mathbb{R}$. For any $\ell \in \mathbb{Z}_{\ge 0}$, it holds that the bit size of the entries of A^ℓ is bounded by $\ell\mathfrak{b}(A) + \ell\mathfrak{b}(n)$ (since each entry is the sum of n^ℓ products of ℓ elements of A).

The matrix equation can be rewritten in the classical vector form: $Mx = N$, where $x = [c_1, \ldots, c_d]$, $N = -\mathrm{vec}(A^d)$ and $M = [\mathrm{vec}(A^{d-1}), \ldots, \mathrm{vec}(A^0)]$ ($\mathrm{vec}(\cdot)$ is the vectorization operator[5]). From the above, it holds that the bitsize of the entries of M is bounded by $n\mathfrak{b}(A) + n^2 \le 2(\mathfrak{b}(A))^2$. Hence, $\mathfrak{b}(M) \le 2(\mathfrak{b}(A))^5$ (since the number of elements of M is equal to $n^2 d$). Similarly, we find that $\mathfrak{b}(N) \le 2(\mathfrak{b}(A))^4$.

Hence, using the algorithm of Proposition 4, we can find integers $e_0 \neq 0$ and $e_1, \ldots, e_d$ such that $A^d + e_1 A^{d-1}/e_0 + \ldots + e_d I/e_0 = 0$ or conclude that no such numbers (integer or not) exist. The smallest d for which such integers exist provides the minimal polynomial of A. Moreover, from the developments above, the bit size of these integers and the time to compute them is polynomial in $\mathfrak{b}(A)$. ■

C. Step 2: Analysis of the roots of a polynomial

The goal of this subsection is to prove Theorem 7 below, which states that deciding whether a polynomial with integer

[4]Again, let us mention that more efficient algorithms have been proposed in the literature (see, e.g., in [8]), but necessitate more work for their description and for the analysis of their complexity. Hence, we present an elementary algorithm to keep the paper simple and self-contained.

[5]I.e., if $A = [a_1, \ldots, a_n] \in \mathbb{R}^{m\times n}$ with $a_i \in \mathbb{R}^m$ for all $i \in \{1, \ldots, n\}$, then $\mathrm{vec}(A) = [a_1^\top, \ldots, a_n^\top]^\top$.

coefficients has the boundedness property can be done in polynomial time w.r.t. the bit size of its coefficients.

The proof relies on the *Routh–Hurwitz stability criterion*, which is an algorithmic test to decide whether all the roots of a given polynomial have negative real part and was shown to be implementable by a polynomial-time algorithm (see, e.g., [14]). Extensions of the Routh–Hurwitz criterion allow to compute for a given polynomial the number of roots on the imaginary axis and their multiplicity (see, e.g., [9, §15], [4], [5]). However, to the best of the authors' knowledge, no proof of the polynomial bit complexity of such algorithms is available in the literature. Hence, in Theorem 7, we present a minimalist version of the extended Routh–Hurwitz algorithm that is sufficient for our needs (verifying the boundedness property), and thriving on Proposition 3, we show that this minimalist version can be implemented by a polynomial-time algorithm.

Theorem 7: There is an algorithm that, given any polynomial $p : x \mapsto a_0x^d + \ldots + a_d$ with integer coefficients $a_0, \ldots, a_d$, outputs "YES" if p has the boundedness property, and outputs "NO" otherwise. Moreover, the bit complexity of the algorithm is polynomial in $\sum_{\ell=0}^{d} \mathfrak{b}(a_\ell)$.

The rest of this section is devoted to proving Theorem 7. To do that, we first introduce several results and concepts that are classical in the study of the Routh–Hurwitz criterion.

Let $p_0, p_1, \ldots, p_{m+1}$ be a sequence of real polynomials such that

$$\left\{ \begin{array}{l} p_{m+1} \equiv 0, \quad p_k \not\equiv 0, \quad \forall\, k \in \{1, \ldots, m\}, \\ p_{k+1} = -\operatorname{rem}(p_{k-1}, p_k), \quad \forall\, k \in \{1, \ldots, m\}, \end{array} \right. \tag{1}$$

where $\operatorname{rem}(p_{k-1}, p_k)$ is the *remainder* of the Euclidean division of p_{k-1} by p_k, meaning that $\deg\, p_{k+1} < \deg\, p_k$ and there is a real polynomial q_k such that $p_{k+1} = q_kp_k - p_{k-1}$. Hence, the polynomials $p_0, \ldots, p_{m+1}$ are those that would be obtained by applying the *Euclidean algorithm* (see, e.g., [6, §1.5]) on p_0 and p_1, which is known to produce the GCD of p_0 and p_1.

Lemma 8 (Folk): Let $p_0, \ldots, p_{m+1}$ be as (1). Then, for all $k \in \{0, \ldots, m\}$, p_m is a greatest common divisor (GCD) of p_k and p_{k+1}.

Proof: See the extended version of this paper [3]. ■

From the above, it holds that p_m divides p_0 and p_1. The following result is known as the *Routh–Hurwitz theorem*.

Lemma 9 (see, e.g., [9, Theorem 15.2]): Let $p_0, \ldots, p_{m+1}$ be as in (1), and let $p : x \mapsto \tilde{p}_0(-ix) + i\tilde{p}_1(-ix)$ where $\tilde{p}_0 = p_0/p_m$ and $\tilde{p}_1 = p_1/p_m$. Then, it holds that

$$\tau_s - \tau_u = V(+\infty) - V(-\infty),$$

where τ_s is the number of roots of p with negative real part and τ_u is the number of roots of p with positive real part, and $V(y)$ $(y \in \mathbb{R} \cup \{\pm\infty\})$ is the number of variations of sign[6] in the sequence $p_0(y), \ldots, p_m(y)$.

Proof: See the extended version of this paper [3]. ■

[6]The *number of variations of sign* in a finite sequence of nonzero real numbers (or $\pm\infty$) is the number of pairs of consecutive elements in the sequence that have opposite sign. For instance, the number of variations of sign in $1, \infty, -1, 3, -\infty, -2$ is equal to 3.

A similar approach can be used to compute the number of distinct real roots of a real polynomial.

Lemma 10 (see, e.g., [9, p. 174]): Let $p_0, \ldots, p_{m+1}$ be as in (1) with $p_1 = -p_0'$. Then, $V(+\infty) - V(-\infty)$ is equal to the number of *distinct real roots* of p_0, where $V(y)$ is the number of variations of sign in $p_0(y), \ldots, p_m(y)$.

Proof: See the extended version of this paper [3]. ■

By combining Lemmas 9 and 10, we obtain the following algorithmic procedure to decide whether a given polynomial has the boundedness property.

To introduce this procedure, let $p : x \mapsto c_0x^d + c_1x^{d-1} + \ldots + c_d$ be a real polynomial ($c_0 \neq 0$). If $\deg\, p$ is even, then decompose p into two real polynomials p_0 and p_1 such that $p(ix) = p_0(x) + ip_1(x)$ for all $x \in \mathbb{C}$. Namely,

$$\begin{array}{l} p_0 : x \mapsto i^dc_0x^d + i^{d-2}c_2x^{d-2} + \ldots + c_d, \\ p_1 : x \mapsto i^{d-2}c_1x^{d-1} + i^{d-4}c_3x^{d-3} + \ldots + c_{d-1}x. \end{array} \tag{2}$$

On the other hand, if $\deg\, p$ is odd, then decompose p into two real polynomials p_0 and p_1 such that $ip(ix) = p_0(x) + ip_1(x)$ for all $x \in \mathbb{C}$. Namely,

$$\begin{array}{l} p_0 : x \mapsto i^{d+1}x^d + i^{d-1}c_2x^{d-2} + \ldots + i^2c_{d-1}x, \\ p_1 : x \mapsto i^{d-1}c_1x^{d-1} + i^{d-3}c_3x^{d-3} + \ldots + c_d. \end{array} \tag{3}$$

In both cases, it holds that $\deg\, p_0 > \deg\, p_1$.

Let $p_0, p_1, \ldots, p_{m+1}$ satisfy (1) with p_0 and p_1 given by (2) or (3). If p_m is not a constant polynomial, then let[7] $p_0^{\text{ext}}, p_1^{\text{ext}}, \ldots, p_{m^{\text{ext}}+1}^{\text{ext}}$ satisfy (1) with $p_0^{\text{ext}} = p_m$ and $p_1^{\text{ext}} = -p_m'$. The following result links the boundedness property with the variations of sign in the sequences $p_0, p_1, \ldots, p_m$ and $p_0^{\text{ext}}, p_1^{\text{ext}}, \ldots, p_{m^{\text{ext}}}^{\text{ext}}$.

Lemma 11: Let $p_0, \ldots, p_{m+1}$ and $p_0^{\text{ext}}, \ldots, p_{m^{\text{ext}}+1}^{\text{ext}}$ be as above. Then, p has the boundedness property if and only if

$$V(+\infty) - V(-\infty) + V^{\text{ext}}(+\infty) - V^{\text{ext}}(-\infty) = \deg\, p_0, \tag{4}$$

where $V(y)$ and $V^{\text{ext}}(y)$ are the number of variations of sign in the sequences $p_0(y), \ldots, p_m(y)$ and $p_0^{\text{ext}}(y), \ldots, p_{m^{\text{ext}}}^{\text{ext}}(y)$.

Proof: See the extended version of this paper [3]. ■

From the above lemma, we obtain the following necessary condition for the satisfiability of the boundedness property.

Corollary 12: Let $p_0, \ldots, p_{m+1}$ and $p_0^{\text{ext}}, \ldots, p_{m^{\text{ext}}+1}^{\text{ext}}$ be as above. A *necessary* condition for p to have the boundedness property is that the degree difference between two consecutive polynomials is equal to one: i.e., $\deg\, p_k = \deg\, p_{k-1} - 1$ for every $k \in \{1, \ldots, m\}$, and $\deg\, p_k^{\text{ext}} = \deg\, p_{k-1}^{\text{ext}} - 1$ for every $k \in \{1, \ldots, m^{\text{ext}}\}$.

Proof: See the extended version of this paper [3]. ■

We are now able to prove Theorem 7 (see below). For this, we use Lemma 11, which requires to compute $V(+\infty) - V(-\infty)$ and $V^{\text{ext}}(+\infty) - V^{\text{ext}}(-\infty)$.[8] We will see that this can be achieved by computing the determinants of matrices

[7]The superscript "ext" stands for "extended" because we extend the sequence $p_0, p_1, \ldots, p_{m+1}$.

[8]The naive way to do this would be to compute the polynomials $p_0, \ldots, p_m$ and $p_0^{\text{ext}}, \ldots, p_{m^{\text{ext}}}^{\text{ext}}$, and look at the variations of sign in the associated sequences. However, a proof of Theorem 7 based on this would require to show that the computation of these polynomials can be done in polynomial time, which is long and tedious (see, e.g., [16, §6]). Therefore, we use another approach, based on the "Hurwitz determinants".

built from the coefficients of p_0 and p_1. This is the idea of the "Hurwitz determinants" obtained from the "Hurwitz matrix" (see, e.g., [9, §15.6]). By combining it with Proposition 3, we deduce that this can be done in polynomial time.

Proof of Theorem 7: First, we explain how to compute $V(+\infty) - V(-\infty)$ and p_0^{ext}; then we apply the exact same idea to compute $V^{\text{ext}}(+\infty) - V^{\text{ext}}(-\infty)$.

Let p_0 and p_1 be as in (2) or (3). For definiteness, suppose that we are in the case of (2) (i.e., the degree of p is even); the case of (3) is exactly the same. Let $d = \deg p = 2f$. If $\deg p_1 < d-1$, then p does not satisfy the necessary condition of Corollary 12 so that there is no need for further computations. Thus, we assume that $\deg p_1 = d-1$. Denote the coefficients of p_0 and p_1 by[9]

$$\begin{aligned} p_0 &: x \mapsto a_0^0 x^d + a_1^0 x^{d-2} + \ldots + a_f^0, \\ p_1 &: x \mapsto a_0^1 x^{d-1} + a_1^1 x^{d-3} + \ldots + a_{f-1}^1 x. \end{aligned} \tag{5}$$

Consider the following $(d+1) \times (d+1)$ matrix:

$$\mathcal{M} = \begin{bmatrix} a_0^0 & a_1^0 & a_2^0 & \cdots & a_f^0 & & & & & \\ & a_0^1 & a_1^1 & \cdots & a_{f-1}^1 & & & & & \\ & a_0^0 & a_1^0 & \cdots & a_{f-1}^0 & a_f^0 & & & & \\ & & a_0^1 & \cdots & a_{f-1}^1 & a_{f-1}^1 & & & & \\ & & a_0^0 & \cdots & a_{f-2}^0 & a_{f-1}^0 & a_f^0 & & & \\ & & & \ddots & & & & \ddots & & \\ & & & & a_0^1 & a_1^1 & a_2^1 & \cdots & a_{f-1}^1 & \\ & & & & a_0^0 & a_1^0 & a_2^0 & \cdots & a_{f-1}^0 & a_f^0 \end{bmatrix}.$$

Let $q_1 : x \mapsto b_1 x$ be such that $\deg(q_1 p_1 - p_0) < \deg p_1$. By (5), it is equivalent to asking that $b_1 a_0^1 - a_0^0 = 0$. Hence, if, for each $k \in \{3, 5, 7, \ldots, d+1\}$, we transform $\mathcal{M}_{[k,:]}$ (the kth row of $\mathcal{M}$) into $b_1 \mathcal{M}_{[k-1,:]} - \mathcal{M}_{[k,:]}$, then we get the following matrix:

$$\mathcal{M}^1 = \begin{bmatrix} a_0^0 & a_1^0 & a_2^0 & a_3^0 & \cdots & a_f^0 & & & & & \\ & a_0^1 & a_1^1 & a_2^1 & \cdots & a_{f-1}^1 & & & & & \\ & & a_0^2 & a_1^2 & \cdots & a_{f-2}^2 & a_{f-1}^0 & & & & \\ & & a_0^1 & a_1^1 & \cdots & a_{f-1}^1 & a_{f-1}^1 & & & & \\ & & & a_0^2 & \cdots & a_{f-2}^2 & a_{f-1}^0 & & & & \\ & & & & \ddots & & & & \ddots & & \\ & & & & & a_0^1 & a_1^1 & a_2^1 & \cdots & a_{f-1}^1 & \\ & & & & & & a_0^2 & a_1^2 & \cdots & a_{f-2}^2 & a_{f-1}^0 \end{bmatrix}.$$

Any kth row of $\mathcal{M}^1$, with $k \in \{3, 5, 7, \ldots, d+1\}$, gives the coefficients of the polynomial p_2, defined by $p_2 = q_1 p_1 - p_0$. Namely, $p_2 : x \mapsto a_0^2 x^{d-2} + a_1^2 x^{d-4} + \ldots + a_{f-1}^2$. The interest of this approach is that to compute the sign of the coefficients $a_0^2, \ldots, a_{f-1}^2$, we do not need to compute $\mathcal{M}^1$, it suffices to compute the determinant of a submatrix of $\mathcal{M}$. More precisely, for all $\ell \in \{0, \ldots, f-1\}$, it holds that

$$\det \mathcal{M}^1_{[1:3,1:2\cup\{\ell+3\}]} = a_0^0 a_0^1 a_\ell^2 = -\det \mathcal{M}_{[1:3,1:2\cup\{\ell+3\}]},$$

since $\mathcal{M}^1_{[1:3,:]}$ was obtained from $\mathcal{M}_{[1:3,:]}$ by using the row transformation $\mathcal{M}^1_{[3,:]} := b_1 \mathcal{M}_{[2,:]} - \mathcal{M}_{[3,:]}$.

In the same way as above, for each $k \in \{4, 6, 8, \ldots, d\}$, we can eliminate the first element of $\mathcal{M}^1_{[k,:]}$ by transforming $\mathcal{M}^1_{[k,:]}$ into $b_2 \mathcal{M}^1_{[k-1,:]} - \mathcal{M}^1_{[k,:]}$, where $q_2 : x \mapsto b_2 x$ is such that $\deg(q_2 p_2 - p_1) < \deg p_1$. This will give a matrix $\mathcal{M}^2$ containing *among others* the coefficients of the polynomial $p_3 : x \mapsto a_0^3 x^{d-3} + a_1^3 x^{d-5} + \ldots + a_{f-2}^3$ defined by $p_3 = q_2 p_2 - p_1$. From this, we get that, for all $\ell \in \{0, \ldots, f-2\}$,

$$a_0^0 a_0^1 a_0^2 a_\ell^3 = \det \mathcal{M}_{[1:4,1:3\cup\{\ell+4\}]}.$$

By using the same reasoning inductively, we get the general relation: for all $k \in \{0, \ldots, d\}$ and $\ell \in \{0, \ldots, f - \lceil k/2 \rceil\}$,

$$a_0^0 \cdots a_0^{k-1} a_\ell^k = \sigma_k \det \mathcal{M}_{[1:k+1,1:k\cup\{\ell+k+1\}]}, \tag{6}$$

where $\sigma_k = -1$ if $k \in 4\mathbb{Z} + 2$ and $\sigma_k = 1$ otherwise, and $p_k : x \mapsto a_0^k x^{d-k} + a_1^k x^{d-k-2} + \ldots + a_{f-\lceil k/2 \rceil}^k x^{k \bmod 2}$ is the kth polynomial in the sequence obtained by (1) with p_0 and p_1 as in (5).

If $\det M_{[1:k+1,1:k+1]} = 0$ for some $k \in \{1, \ldots, d\}$, this means that $a_0^k = 0$. In this case, two situations can occur:

(S1) $\det \mathcal{M}_{[1:k+1,1:k\cup\{\ell+k+1\}]} = 0$ for all $\ell \in \{1, \ldots, f - \lceil k/2 \rceil\}$. This means that $p_k \equiv 0$.

(S2) $\det \mathcal{M}_{[1:k+1,1:k\cup\{\ell+k+1\}]} \neq 0$ for some $\ell \in \{1, \ldots, f - \lceil k/2 \rceil\}$. This means that $p_k \not\equiv 0$ but $\deg p_k < \deg p_{k-1} - 1$.

The above leads to the following algorithm for the computation of $V(+\infty) - V(-\infty) + V^{\text{ext}}(+\infty) - V^{\text{ext}}(-\infty)$.

Algorithm: Using Proposition 3, we compute the determinant of $\mathcal{M}_{[1:k+1,1:k+1]}$ for $k = 0, 1, \ldots, d$. If the determinant is nonzero for all $k \in \{0, \ldots, d\}$, then we deduce from (6) the signs of the leading coefficients $a_0^0, \ldots, a_0^d$. We verify whether these signs are strictly *alternating* since this is the only way to have $V(+\infty) - V(-\infty) = d$. If this is the case, we stop the algorithm and output "YES" since p satisfies (4) in Lemma 11. Otherwise, we stop the algorithm and output "NO" since p does not satisfy (4) in Lemma 11.

On the other hand, if, for some $k \in \{1, \ldots, d\}$, the determinant of $\mathcal{M}_{[1:k+1,1:k+1]}$ is zero, then we check whether we are in situation (S1) or (S2) above. If we are in (S2), then it means that p does not satisfy the necessary condition of Corollary 12, and thus, we stop the algorithm and output "NO". Otherwise (we are in (S1)), we let m be the smallest k such that $\det \mathcal{M}_{[1:k+2,1:k+2]} = 0$, and from (6) we compute the sign of $a_0^0, \ldots, a_0^m$ (the leading coefficients of $p_0, \ldots, p_m$). If the signs are not strictly *alternating*, then we stop the algorithm and output "NO" since p does not satisfy (4) in Lemma 11. Otherwise, from (6) with $k = m$ and $\ell = 0, \ldots, f - \lceil m/2 \rceil$, we define $p_0^{\text{ext}} = |a_0^0 \cdots a_0^{m-1}| p_m$.

At this stage, if the algorithm did not stop and outputted "Yes" or "NO", then it produced the polynomial $p_0^{\text{ext}} = \alpha p_m$ with $\alpha = |a_0^0 \cdots a_0^{m-1}| > 0$. From p_0^{ext}, we compute the value of $V^{\text{ext}}(+\infty) - V^{\text{ext}}(-\infty)$ as follows. First, we define $p_1^{\text{ext}} = -(p_0^{\text{ext}})'$. Then, in the same way as above, we compute the signs of $a_0^{\text{ext},0}, \ldots, a_0^{\text{ext},m^{\text{ext}}}$ (the leading coefficients of $p_0^{\text{ext}}, \ldots, p_{m^{\text{ext}}}^{\text{ext}}$). If $m^{\text{ext}} = d - m$ and the signs are strictly

[9]In the rest of this subsection, for the sake of readability, superscripts are used both as exponents and as indexes, but the distinction should be clear from the context; e.g., a_0^0, $\mathcal{M}^1$ (*index*) vs. x^d (*exponent*).

```
Input: $A = B/q$, with $A \in \mathbb{Z}^{n\times n}$ and $q \in \mathbb{Z}_{>0}$.
Output: "YES" if $A$ is a positive instance of Problem 2 and "NO" otherwise.
Algorithm:
▷ Step 1: Using Theorem 6, compute integers $e_0 \neq 0$ and $e_1, \ldots, e_d$ such that $x \mapsto x^d + e_1x^{d-1}/e_0 + \ldots + e_d/e_0$ is the minimal polynomial of $B$.
Let $\hat{p} : x \mapsto e_0q^dx^d + e_1q^{d-1}x^{d-1} + \ldots + e_1q + e_d$.
▷ Inter-step: Using Theorem 14, compute a polynomial $p$ that has the boundedness property if and only if $\hat{p}$ has the discrete-time boundedness property.
▷ Step 2: Using Theorem 7, return "YES" if $p$ has the boundedness property and return "NO" otherwise.
```

Fig. 3. Algorithm for answering Problem 2.

alternating, then we stop the algorithm and output "YES" since p satisfies (4) in Lemma 11. Otherwise, we stop the algorithm and output "NO" since p does not satisfy (4) in Lemma 11. ◁

The above algorithm requires at most d^2 computations of the determinant of a submatrix of $\mathcal{M}$. By Proposition 3, these determinants can be computed in polynomial time w.r.t. the bit size of the entries of $\mathcal{M}$. Since these entries consist in the coefficients of the input polynomial p, this concludes the proof of the theorem. ■

V. Proof of Theorem 2

The polynomial-time algorithm to answer Problem 1 presented in the previous section (see Figure 2) can be easily adapted, by adding an intermediate step between Step 1 and Step 2, to obtain a polynomial-time algorithm for Problem 2. The intermediate step consists in a transformation of the minimal polynomial of the matrix, called a *Möbius transformation*, which maps the interior of the unit circle in the complex plane to the interior of the left-hand side plane. The relevance of this transformation is explained in Theorem 13 below.

Definition 3: A (complex or real) polynomial will be said to have the *discrete-time boundedness property* if each of its roots satisfies one of the following two conditions: (i) is in the interior of the unit circle, or (ii) is on the unit circle and is simple.

Theorem 13 (Folk): For any $A \in \mathbb{R}^{n\times n}$, it holds that $\sup_{t\in\mathbb{Z}_{\geq 0}} \|A^t\| < \infty$ if and only if the minimal polynomial of A has the discrete-time boundedness property.

Proof: See the extended version of this paper [3]. ■

Theorem 14: There is an algorithm that, given any polynomial $\hat{p} : x \mapsto a_0x^d + \ldots + a_d$ with integer coefficients $a_0, \ldots, a_d$, outputs a polynomial p such that $\hat{p}$ has the discrete-time boundedness property if and only if p has the boundedness property. Moreover, the bit complexity of the algorithm is polynomial in $\sum_{\ell=0}^{d} \mathfrak{b}(a_\ell)$.

Proof: See the extended version of this paper [3]. ■

Putting things together, we get the polynomial-time algorithm presented in Figure 3 to answer Problem 2.

VI. Conclusions

Summarizing, in this paper, we showed that the problem of deciding whether a linear time invariant dynamical system, with rational transition matrix, has bounded trajectories can be answered in polynomial time with respect to the bit size of the entries of the transition matrix. To do this, we leveraged several tools from system and control theory and from computer algebra, and we provided a careful analysis of the computational complexity of these tools when integrated into a complete algorithm for our decision problem.

For further work, it would interesting to derive tight upper bounds on the complexity of the described algorithm (and of some improved versions not presented here to keep the paper simple and self-contained), and also to compare it with the complexity that could be obtained with other types of algorithms, like randomized algorithms, which are known to provide practically efficient algorithms, for instance, for the computation of the determinant of integer matrices, or for the computation of the GCD of polynomials with integer coefficients.

References

[1] Erwin H Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Mathematics of Computation*, 22:565–578, 1968.

[2] Richard H. Bartels and George W Stewart. Algorithm 432: solution of the matrix equation $AX + XB = C$. *Communications of the ACM*, 15(9):820–826, 1972.

[3] Guillaume O Berger and Raphaël M Jungers. Complexity of the lti system trajectory boundedness problem. *arXiv preprint arXiv:2108.00728*, 2021.

[4] Shyan S Chen and Jason SH Tsai. A new tabular form for determining root distribution of a complex polynomial with respect to the imaginary axis. *IEEE transactions on automatic control*, 38(10):1536–1541, 1993.

[5] Mohammad Amin Choghadi and Heidar A Talebi. The routh-hurwitz stability criterion, revisited: the case of multiple poles on imaginary axis. *IEEE Transactions on Automatic Control*, 58(7):1866–1869, 2013.

[6] David A Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer, Cham, 4th edition, 2015.

[7] Jean-Guillaume Dumas. Bounds on the coefficients of the characteristic and minimal polynomials. *arXiv preprint cs/0610136*, 2006.

[8] Jean-Guillaume Dumas, Clément Pernet, and Zhendong Wan. Efficient computation of the characteristic polynomial. In *Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation*, pages 140–147. ACM, 2005.

[9] Felix R Gantmacher. *The theory of matrices, Vol. 2*. American Mathematical Society, Providence, RI, 2000.

[10] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, MA, 2nd edition, 2013.

[11] Adolf Hurwitz. Ueber die Bedingungen, unter welchen eine Gleichung nur Wurzeln mit negativen reellen Teilen besitzt. *Mathematische Annalen*, 46:273–284, 1895.

[12] Alfred-Marie Liénard and Henri Chipart. Sur le signe de la partie réelle des racines d'une équation algébrique. *Journal de Mathématiques Pures et Appliquées*, 10:291–346, 1914.

[13] Douglas Lind and Brian Marcus. *An introduction to symbolic dynamics and coding*. Cambridge University Press, Cambridge, UK, 1995.

[14] Juan M Peña. Characterizations and stable tests for the Routh–Hurwitz conditions and for total positivity. *Linear Algebra and its Applications*, 393:319–332, 2004.

[15] Edward John Routh. *A treatise on the stability of a given state of motion*. Macmillan, London, 1877.

[16] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge University Press, New York, NY, 3rd edition, 2013.