CIRCUITS SYSTEMS SIGNAL PROCESSING
VOL. 23, NO. 3, 2004, PP. 195–213

AN EFFICIENT INTEGER-PRESERVING STABILITY TEST FOR DISCRETE-TIME SYSTEMS*

Yuval Bistritz[1]

Abstract. The paper presents an efficient integer-preserving version for the author's stability test for discrete-time linear systems. A first naive solution that satisfies this constraint is shown to have an explosive (severely exponential) growth of the magnitude of the integers. Then a simple, but far from obvious, new recursion form is established that has a more restrained (linear) growth of coefficients. A qualitative evaluation of computing time shows that the new test form is most efficient. Its possible usefulness for determining stability constraints for filters and systems with designable parameters is illustrated by a numerical example. Its capacity to offer better numerical accuracy for high-degree polynomials is also illuminated. Additional applications may arise from its usability over other algebraic rings. The latter capacity was demonstrated recently by implementing it into an efficient stability test for two-dimensional discrete-time systems.
Key words: Stability criteria for discrete-time systems, unit-circle zero location, immittance algorithms, integer-preserving computation.

1. Introduction

Algebraic algorithms to test whether a polynomial has all its zeros inside the unit circle (i.e., is "stable") play a central role in digital signal processing and discrete-time system theory. They are used to test stability and to design stable filters and systems. They are also related to additional processing algorithms and filter forms. Thus new forms of one-dimensional (1D) stability tests made impact on associated signal processing algorithms and may also affect corresponding algorithms for higher dimensional systems.

The classical Schur–Cohn and Marden–Jury (SCMJ) stability tests [3] rely on two-term polynomial recursion that propagates (asymmetric) polynomials of decreasing degree that are also common to the Levinson algorithm for linear

* Received June 23, 2003; revised December 1, 2003.
[1] Department of Electrical Engineering, Tel Aviv University, Tel Aviv 69978 Israel. E-mail: bistritz@eng.tau.ac.il

prediction and the ParCor lattice filters. An alternative formulation for these algorithms emerged from the author's stability test in [2]. The new formulation relies on a three-term recursion that propagates polynomials with symmetry. It has become termed the *immittance* (or "split") formulation and, for distinction, the classical algorithms were given the label *scattering* algorithms. The immittance algorithms exploit intrinsic symmetry in the underlying problems to solve the respective classical algorithms with less computation; see, e.g., [8]. A chapter on scattering versus immittance algorithms appears in the textbook [13, Chap. 5].

The mentioned stability test [2], a refined version of which appeared recently in [5], has become reputable as the least cost stability test (and unit-circle zero location procedure) for one-dimensional discrete-time systems test (1D stability test) in a conventional count of operations. However, there exist applications for which considerations beyond minimal count of operations become an important issue. 1D stability tests are used at times as a tool to design filters or to stabilize a system by feedback. These applications could benefit from a 1D stability test that exhibits the property that, if the tested polynomial has integer coefficients, then the test can be completed with all operations performed over integers only, a property to which we shall refer as the integer-preserving (IP) property.

This paper proposes an efficient IP form for the test in [2]. The scope of interest in such an algorithm is evidently not limited to integer polynomials. A polynomial recursion that preserves coefficients over integers (i.e., is not moving to rational numbers) also preserves coefficients over any algebraic ring (i.e., is not moving to its respective quotient field). For instance, the coefficients might belong to the ring of real-coefficient polynomials (in one or several variables) or the ring of integer-coefficient polynomials. A situation where the coefficients are themselves polynomials arises in testing multidimensional discrete-time systems. Indeed, the stability conditions and recursion to be derived here were already exploited to develop some efficient two-dimensional (2D) stability tests (tests for stability of two-dimensional discrete-time systems) in [6], [7]. Another area of application is the use of such a test as a design tool in a symbolic programming environment. A numerical example in this paper will illustrate an instance of this kind of application. The usefulness of the IP algorithm in increasing the numerical accuracy is also anticipated because manipulation of binary numbers in computers resembles a power-of-two ring of polynomials. In view of the aforementioned interrelations between stability tests and other signal processing algorithms, the proposed test may also affect algorithms beyond the stability issue.

There already exists a scattering-type 1D stability that is integer preserving. A modified test that Jury proposed on several occasions in various versions [10], [11], [12] actually has this property [1]. (The various versions of the modified Jury test (MJT), their interrelations, and some of their properties are summarized under the C-type category in the classification of the SCMJ tests into four types in [3].) Jury arrived to this modification through a search for a stability test that directly produces the principal minors of the Schur–Cohn matrix. This interest was in

turn motivated by its usefulness in developing stability tests for 2D discrete-time systems. The MJT led to the very efficient 2D stability test [9], [4]. The test in this paper will be shown to be more efficient than this alternative IP stability test.

An obvious approach to achieve integer preservation is to avoid the division operation. This approach leads to a division-free stability test, which is explored in Section 2. Its analysis is shown to lead to a very rapid (exponential) growth of coefficient lengths. Section 3 identifies the rapid growth of coefficient as caused by certain common factors that the division-free algorithm creates and magnifies from step to step of the recursion.

Section 3 then discusses a very simple, but not obvious, modification of the division-free recursion that restrains the coefficients' growth. The modified recursion brings back the division operation, but now it acts to extract exact integer factors that cause the inflated coefficients' growth. Consequently, the modified algorithm remains integer preserving. Necessary and sufficient conditions for stability are derived for the modified recursion. This fraction-free stability test is shown to have only linear growth of coefficients.

Section 4 examines the efficiency of the test in terms of count of integer (or "binary") operations. The test is shown to exhibit an efficiency that exceeds by a large factor the corresponding efficiency of the MJT, the only other available IP stability test for discrete systems.

The paper focuses essentially only on testing the stability of a real polynomial. A desirable extension to a general IP unit-circle zero location procedure will not be considered. However, zero location rules for polynomials that can be run by the recursion without requiring changes in its normal form can be obtained in a reasonably straightforward manner from respective rules in [2] and will therefore be presented. Numerical examples are given for both the division-free and the fraction-free form tests to illustrate their behavior as stability tests and, for the fraction-free test, also to illustrate its use as a zero location procedure and to determine intervals of stability for a symbolic parameter.

2. Division-free test

We begin by presenting as the following algorithms and theorem the stability test in [2] or [5] for real polynomials. This and some subsequent comments will be helpful for the following developments and will provide the reader independence from reading a paper on the original test form. We shall use $\mathbb{I}$, $\mathbb{Q}$, and $\mathbb{R}$ to denote the ring of integers, the field of rational numbers, and the field of real numbers, respectively, and $\mathbb{I}[z]$, $\mathbb{Q}[z]$, and $\mathbb{R}[z]$ will denote the set of polynomials with coefficients in the respective domains.

Assume a polynomial

$$D_n(z) = \sum_{i=0}^{n} d_i z^i \in \mathbb{R}[z], \qquad D_n(1) > 0,\ d_n \neq 0. \tag{1}$$

Let the reciprocal of a polynomial $D_n(z)$ be defined and denoted by $D_n^\sharp(z) := \sum_{i=0}^n d_{n-i} z^i$. A polynomial is called symmetric if $D_n(z) = D_n^\sharp(z)$. (The term "reciprocal" stems from the property that the zeros of $D_n^\sharp(z) = z^n D(z^{-1})$ are the reciprocal of the zeros of $D_n(z)$. Calling a polynomial symmetric refers to the fact that its coefficients exhibit symmetry with respect to their center, i.e., $d_{n-i} = d_i$, $i = 0, \ldots, n$. A real polynomial is either symmetric or antisymmetric if and only if it is self-reciprocal.)

Algorithm 1: *Original Form*. *Consider $D_n(z)$ (1) and assign to it a sequence of $n+1$ symmetric polynomials $\{T_m(z) = \sum_{i=0}^m t_{m,i} z^i,\ m = n, n-1, \ldots, 0\}$ as follows:*

$$T_n(z) = D_n(z) + D_n^\sharp(z), \qquad T_{n-1}(z) = \frac{D_n(z) - D_n^\sharp(z)}{(z-1)}. \tag{2}$$

For $m = n-1, \ldots, 1$ do:

$$zT_{m-1}(z) = \delta_{m+1}(z+1)T_m(z) - T_{m+1}(z), \qquad \delta_{m+1} = \frac{t_{m+1,0}}{t_{m,0}}. \tag{3}$$

Theorem 1. *A polynomial $D_n(z)$ (1) is stable (i.e., if $D_n(z_i) = 0$, then $|z_i| < 1$) if and only if Algorithm 1 yields for it*

$$T_m(1) > 0 \qquad m = n, \ldots, 0. \tag{4}$$

If $D_n(z)$ is stable, then the following condition holds as well:

$$t_{m,0} > 0 \qquad m = n, \ldots, 0. \tag{5}$$

Normal conditions for Algorithm 1 are defined as

$$t_{m,0} \neq 0, \qquad m = n, \ldots, 0. \tag{6}$$

If we call a polynomial like $D_n(z)$ normal if its leading coefficient $d_n \neq 0$ and abnormal when $d_n = 0$, then normal conditions present the situation where all $T_m(z)$ are normal. It is apparent that an abnormal $T_m(z)$ interrupts the normal recursion. In such a case not all the terms required in (4) are defined. To avoid such a concern, the second part of the theorem states that normal conditions are necessary conditions for stability. Thus, normal conditions provide a sufficient framework for using the procedure as a stability criterion.

Remark 1. Two noteworthy clarifications on the assumptions made in (1) are as follows. The condition $D_n(1) > 0$ implies that the test should start by examining $D_n(1)$. If $D_n(1) = 0$, then $D_n(z)$ is not stable and the test terminates (when the purpose is to find the distribution according to [2], the zeros at $z = 1$ can first be factored out). A $D_n(z)$ such that $d_n = 0$ is again not stable as it has a zero outside the unit circle at ∞. (Again if zero location is sought, the zero location test can proceed with the lower degree normal polynomial.) The second clarification regards the assumption in (1) that the sign of $D_n(1)$ is positive. First, note that, for a stable polynomial, d_n and $D_n(1)$ have the same sign. (This is

realized by setting $z = 1$ into $D_n(z) = d_n \prod_{i=1}^{n}(z - z_i)$.) Thus, the pair of conditions $d_n > 0$ and $D_n(1) \neq 0$ is interchangeable with the pair $d_n \neq 0$ and $D_n(1) > 0$ used in (1). Most often, the sign assumption in (1) is not restrictive because if it does not hold (for a normal $D_n(z)$ that does not vanish at $z = 1$), the stability of the negated polynomial $-D_n(z)$ can be tested instead. However, on some occasions the test might be embedded in applications where a change of sign is not permitted. It turns out that staying uncommitted to the sign of $D_n(1)$ or d_n (i.e., requiring only that they do not vanish) complicates unnecessarily the manner in which subsequent theorems have to be stated and proved. We therefore will present theorems and prove them assuming the positivity condition as in (1) and afterward will remark on the changes that occur in the complementary case of negative sign.

Suppose now that Algorithm 1 is applied to a $D_n(z) \in \mathbb{I}(z)$. Obviously, the first two polynomials are also $T_n(z)$, $T_{n-1}(z) \in \mathbb{I}[z]$ but subsequently, for $m \leq n - 2$, $T_m(z) \in \mathbb{Q}[z]$ in general. In other words, in general Algorithm 1 will test a $D_n(z) \in \mathbb{I}(z)$ over the field of rational numbers. Clearly, the departure to rational coefficients is caused by the division operation contained in the δ_m's. Hence, an obvious way to ascertain that all $T_m(z) \in \mathbb{I}[z]$ is to avoid divisions. The remainder of this section explores a division-free stability test based on this simple idea.

Algorithm 2: *Division-Free Form.* *Assume $D_n(z)$ (1) and assign to it a sequence of $n + 1$ symmetric polynomials $\{F_m(z) = \sum_{i=0}^{m} f_{m,i} z^i,\ m = n, \ldots, 0\}$ as follows:*

$$F_n(z) = D_n(z) + D_n^{\sharp}(z), \qquad F_{n-1}(z) = \frac{D_n(z) - D_n^{\sharp}(z)}{(z-1)}. \tag{7}$$

For $m = n - 1, \ldots, 1$ do:

$$zF_{m-1}(z) = f_{m+1,0}(z+1)F_m(z) - f_{m,0}F_{m+1}(z). \tag{8}$$

It is clear that Algorithm 2 produces for a $D_n(z) \in I[z]$ a sequence of symmetric polynomials such that $F_m(z) \in I[z]$ for all $m = n, \ldots, 0$, i.e., that it is integer preserving. Normal conditions are again defined as the case when all the produced symmetric polynomials are normal, viz.,

$$f_{m,0} \neq 0, \qquad m = n, \ldots, 0. \tag{9}$$

This time an $f_{m-1,0} = 0$, $m < n$ does not cause division by zero, but it still obstructs the degree reduction mechanism and prevents the completion of Algorithm 2. Normal conditions provide a sufficient framework for Algorithm 2 to produce the complete sequence $\{F_m(z) = \sum_{i=0}^{m} f_{m,i} z^i,\ m = n, \ldots, 0\}$ and, as will be shown shortly, a sufficient framework for testing stability.

Lemma 2. *For a given $D_n(z)$, Algorithm 2 is normal if and only if Algorithm 1*

is normal. Assuming normal conditions, the respective polynomials are proportional,

$$F_m(z) = \psi_m T_m(z), \tag{10}$$

where the scalars $\psi_m \in \mathbb{R}$ *are given by* $\psi_n = 1$, $\psi_{n-1} = 1$ *and then (when the sequence of* $\{T_m(z)\}$ *is given) can be computed recursively by*

$$\psi_m = \psi_{m+2}\psi_{m+1}t_{m+1,0}, \qquad m \le n-2 \tag{11}$$

or (when the sequence of $\{F_m(z)\}$ *is given) by*

$$\psi_m = \psi_{m+2} f_{m+1,0}, \quad m \le n-2. \tag{12}$$

Proof. Set (10) and its consequence $f_{m,0} = \psi_m t_{m,0}$ into the right-hand side of (8),

$$\begin{aligned} zF_{m-2}(z) &= \psi_m t_{m,0}(z+1)\psi_{m-1}T_{m-1}(z) - \psi_{m-1}t_{m-1,0}\psi_m T_m(z) \\ &= \psi_m \psi_{m-1} t_{m-1,0}\left[\frac{t_{m,0}}{t_{m-1,0}}(z+1) - T_m(z)\right] \\ &= \psi_m \psi_{m-1} t_{m-1,0} z T_{m-2}(z) \end{aligned}$$

to obtain (11). Use again $f_{m,0} = \psi_m t_{m,0}$ to conclude (12). □

Theorem 3. *A polynomial* $D_n(z)$ *(1) is stable if and only if Algorithm 2 yields for it*

$$F_m(1) > 0 \qquad m = n, n-1, \ldots, 0. \tag{13}$$

If $D_n(z)$ *is stable, then the following condition holds as well:*

$$f_{m,0} > 0 \qquad m = n, n-1, \ldots, 0. \tag{14}$$

Proof. If $D_n(z)$ is stable, then Theorem 1 implies that all $t_{m,0} > 0$. Therefore, by (11), all $\psi_m > 0$. Therefore, (10) implies (13) and then (12) implies (14). To show the converse, denote $\varphi_m := F_m(1)$ and set $z = 1$ into (8) to obtain the recursion

$$\varphi_{m-1} = 2f_{m+1,0}\varphi_m - f_{m,0}\varphi_{m+1}. \tag{15}$$

Assume that (13) holds, i.e., all $\varphi_m > 0$. Then $f_{0,0} = F_0(1) > 0$, $f_{1,0} = F_1(1)/2 > 0$ proceed with the recursion (15) upward with indices for $m = 1, 2, \ldots, n-1$ to show, step after step, that $f_{m+1,0} > 0$. This proves that (13) implies (14). The latter implies that $\psi_m > 0$ for all m via (12). Then use (10) to also conclude (4), which in turn offers sufficient conditions for the stability of $D_n(z)$ according to Theorem 1. □

Note that the recursion (15) initiated with $\varphi_n = F_n(1)$ $\varphi_{n-1} = F_{n-1}(1)$ can be added to Algorithm 2 and used to obtain more efficiently the values required for (13).

Remark 2. If $D_n(1) < 0$ (instead of the assumption in (1)), then the signs in both (13) and (14) have to be changed from all positive to the following pattern $-, -, +, -, -, +, \ldots$ for $m = n, n-1, \ldots$. For example, the necessary and sufficient

conditions for stability are $F_n(1) < 0$, $F_{n-1}(1) < 0$, $F_{n-2}(1) > 0$, $F_{n-3}(1) < 0$, $F_{n-4}(1) < 0$, $F_{n-5}(1) > 0$, etc.

Example 1. Consider testing the polynomial $D_7(z) = 1 + 3z + 2z^2 + 4z^3 + 8z^4 + 7z^5 + 5z^6 + 8z^7$ by the division-free test. Algorithm 2 yields the next table, arranged as in [2]. Namely, the rows that correspond to the coefficients of successive polynomials of the sequence and entries available by symmetry are put in parentheses (and also partly truncated to save space).

9		8		9		12		(12)		...
	7		9		14		18		(14)	...
		88		144		204		(204)		...
			832		1204		1272		(1204)	...
				59360		48160		(48160)		...
					17987200		4632320		(17987200)	
						476431155200		(476431155200)		
							14932343380770816000			

The integer preservation property already acquires the division-free test with some immunity to roundoff error for integer polynomials (by using integer arithmetics or rounding to integers numbers in a floating-point environment). However, as the preceding relatively low-degree polynomial already hints, the major drawback of the algorithm is a very rapid growth of the integers.

To characterize the growth of coefficients, one needs some measure for the length of coefficients. Some possibilities are the number of decimal digits or the number of bits required for the presentation of the integer (or, say, the log on base 10 or 2 of the number rounded to integers). Any of these measures is equally qualifying and has the properties that the product of two integers of lengths b_1 and b_2 has length less than or equal to $b_1 + b_2$, and that their sum or difference has length less than or equal to $\max(b_1, b_2) + 1$. (As a matter of fact, we need not make any specific choice among them for the sake of our qualitative analysis.) Let B characterize the length of coefficients of $D_n(z) \in \mathbb{I}(z)$ (say, we characterize the length of coefficients of a polynomial by its longest coefficient). Then the length of the coefficients of $T_n(z), T_{n-1}(z) \in I(z)$ is at most $B+1$ and $B+2$, respectively (they are formed by one and two passes, respectively, of additive operations). For further simplicity of the estimate in the following proposition, it will be assumed that B presents the lengths of the coefficients of $D_n(z)$ and also of $T_n(z)$ and $T_{n-1}(z)$.

Theorem 4. *The coefficients' length in Algorithm 2 grows at an exponential rate. With B presenting the length of the coefficients of $D_n(z) \in I(z)$ (and also of $F_n(z)$ and $F_{n-1}(z)$ as explained above), the lengths for $F_{n-k}(z)$ for $k = 0, 1 \ldots, n$ are approximately given by* $\ell_f(k) = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{(k+1)} - \left(\frac{1-\sqrt{5}}{2}\right)^{(k+1)}\right]B$.

Proof. Let $\ell_f(k)$ denote a bound on the length of the coefficients of $F_{n-k}(z) \in \mathbb{I}[z]$ produced by Algorithm 2. It is apparent from (8) that $\ell_f(m)$ evolves as a Fibonacci sequence $\ell_f(k) = \ell_f(k-1) + \ell_f(k-2)$ for $k = 2, \ldots, n$. The

solution of this difference equation for the assumed initial conditions $\ell_f(k) = B$ for $k = 0, 1$ is $\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{(k+1)} - \left(\frac{1-\sqrt{5}}{2}\right)^{(k+1)}\right]B$. □

Evidently, the mode $\lambda_o := \frac{1+\sqrt{5}}{2} > 1$ causes an exponential growth of the integers' length as a function of n. This exponential growth impairs the use of the division-free stability test for already moderate-degree polynomials that arise in practical application. It may create integers that exceed the computer wordlength (or in multiprecision mode pose exhaustive demands on memory). Using it with floating-point arithmetic, the rapid growth of coefficients would deteriorate accuracy by rounding error. Other generalized applications like using it to determine stability intervals for parameters also suffer from such rapid growth of coefficients. The following example illustrates such an application.

Example 2. Suppose that in Example 1 we leave an indeterminate as follows: $D_7(z; K) := K + 3z + 2z^2 + 4z^3 + 8z^4 + 7z^5 + 5z^6 + 8z^7$ and that the problem is to determine an interval of K for which $D_7(z; K)$ is stable.

First one has to construct the sequence $F_m(z; K)$ using Algorithm 2. We skip the resulting polynomials. Then setting into them $z = 1$ yields

$$
\begin{aligned}
F_7(1) &= 74 + 2K \\
F_6(1) &= 85 - 7K \\
F_5(1) &= 768 + 116K - 12K^2 \\
F_4(1) &= 5488 + 30K - 184K^2 + 10K^3 \\
F_3(1) &= 214528 + 17408K - 16920K^2 - 304K^3 + 344K^4 - 16K^5 \\
F_2(1) &= 57405440 - 8471040K - 9479616K^2 + 789768K^3 + 407256K^4 - 40832K^5 \\
&\quad -4992K^6 + 760K^7 - 24K^8 \\
F_1(1) &= 1760978534400 - 328639447040K - 550548275200K^2 + 27063975936K^3 \\
&\quad +48297764864K^4 - 2520729344K^5 - 1967714944K^6 + 166673408K^7 + 36661568K^8 \\
&\quad -5056448K^6 - 127488K^7 + 53888K^8 - 3264K^9 + 64K^{10} \\
F_0(1) &= 2814860792011161600 - 2600163429947801600K - 125758100165623808000K^2 \\
&\quad +15948935214399488 0K^3 + 1988975345742643200K^4 - 34102243524345856K^5 \\
&\quad -170456022185803776K^6 + 7946796620759040K^7 + 8869929111477248K^8 \\
&\quad -7803141153612 80K^9 - 276734510590464K^{10} + 39586114907904K^{11} \\
&\quad +4285091708928K^{12} - 1091853806848K^{13} + 5154840576K^{14} + 14569554432K^{15} \\
&\quad -1163694080K^{16} - 37160960K^{17} + 10955264K^{18} - 717056K^{19} + 21504K^{20} - 256K^{21}
\end{aligned}
$$

Stability intervals for K occur when the preceding set of polynomials satisfies the conditions in Theorem 2. Namely, one has to find values of K for which all these polynomials are commonly positive. Note that the degrees in K attain the bounds prescribed in Theorem 4 until $\ell(7) = 21$ (where "length" is measured in powers of K and initially the length for K is bounded by its degree $B = 1$ in $D_7(z; K)$). We shall not solve this problem for now but will get back to it later, in Example 4, solving it instead with the proposed procedure.

Note that solving the problem in Example 2 using the original stability test is

not easy either. It would lead to inequalities posed on $n+1=8$ rational functions of K, which means examining pairs of polynomials in K (with numerator and denominator of degrees of up to $n = 7$) for being either both positive or both negative.

The division-free algorithm also raises difficulties in other potential applications for a stability test over other integral ("integer-like") domains. For example, using it for testing the stability of 2D systems has a vast cost of computation [6]. The next section brings a very simple and effective remedy to the rapid growth of coefficients length.

3. Reducing coefficients' growth

This section first shows that the rapid growth of coefficients in the division-free recursion is caused by the fact that Algorithm 2 produces polynomials with coefficients that contain common factors of elements of their domain of definition and it tends to pile up these common factors. A modification of the recursion into a fraction-free algorithm that eliminates these common factors in a systematic and simple manner is obtained. Stability conditions (as well as the zero location rule in normal conditions) for the fraction-free algorithm are derived afterwards. Finally, the growth of coefficients of the fraction-free test is shown to be only linear.

3.1. Redundant common factors

The next lemma shows and characterizes the fact that applying the division-free algorithm to $D_n(z) \in \mathbb{I}[z]$ creates and piles up integer factors common to all the coefficients of each $F_m(z)$.

Lemma 5. *Consider the sequence* $\{F_m(z)\}_0^n$ *produced by Algorithm 2 for a* $D_n(z) \in \mathbb{I}[z]$.

(a) Let $G_i(z) = F_{m-i}(z), i = 0, 1, 2, 3, 4$ *denote consecutive polynomials* $G_i(z) = \sum g_{i,0} z^i$ *in this sequence, for any* $n \geq m \geq 4$. *Then* $g_{1,0}$ *divides* $G_4(z)$, *i.e.,* $G_4(z)/g_{1,0} \in \mathbb{I}[z]$.

(b) 2 divides $F_{n-2}(z)$, *i.e.,* $\frac{1}{2}F_{n-2}(z) \in \mathbb{I}[z]$, *and its elimination does not intervene with the mechanism described in (a).*

(c) If $f \in \mathbb{I}$ *divides* $F_m(z)$, *i.e.,* $F_m(z)/f \in \mathbb{I}[z]$, *for any* $m \leq n-1$, *then* f *also divides all subsequent polynomials as well, i.e.,* $F_k(z)/f \in \mathbb{I}[z]$ *also for* $k = m-1, \ldots, 0$.

Proof. The proof of claim (a) can be completed by carrying out the implied sequence of substitution. Denote the coefficients of each $G_i(z)$ as a row vector,

$$G_i = [g_{i,0},\ g_{i,1},\ \ldots,\ g_{i,1}\ ,\ g_{i,0}].$$

Starting with $i = 0, 1$, the recursion forms

$$\begin{aligned}[0, G_2, 0] &= g_{0,0}\{\,[G_1, 0] + [0, G_1]\,\} - g_{1,0}G_0 \\ &= [0\ ,\ g_{2,0}\ ,\ g_{2,1}, \ldots, g_{2,1}\ ,\ g_{2,0}\ ,\ 0]\end{aligned} \tag{16}$$

$$\begin{aligned}[0, 0, G_3, 0, 0] &= g_{1,0}\{\,[0, G_2, 0, 0] + [0, 0, G_2, 0]\,\} - g_{2,0}[0, G_1, 0] \\ &= [0\ ,\ 0\ ,\ g_{3,0}\ ,\ g_{3,1}\ ,\ \ldots\ ,\ g_{3,1}\ ,\ g_{3,0}\ ,\ 0\ ,\ 0]\end{aligned} \tag{17}$$

$$[0, 0, 0, G_4, 0, 0, 0] = g_{2,0}\{\,[0, 0, G_3, 0] + [0, G_3, 0, 0]\,\} - g_{3,0}[0, 0, G_2, 0, 0], \tag{18}$$

where the zeros are padded to match vanishing coefficients that the degree-reducing recursion introduces. It is possible to alleviate somewhat the awkward process of backward substitution until G_4 is expressed by the claimed common factor by using an insight in to the nature of the recursion as follows. An inherent (and readily verified) property of the recursion is that a term on the right-hand side sum that is divisible by a certain factor (where we shall be focusing on $g_{1,0}$) contributes to the linear combination of terms that composes future polynomials, terms that also contain that factor. This property will be used to drop in a sum of terms, terms that are already seen to contain the factor $g_{1,0}$. Begin by reading from (16) that

$$g_{2,0} = g_{0,0}g_{1,1} + g_{0,0}g_{1,0} - g_{1,0}g_{0,1} \mapsto g_{0,0}g_{1,1}, \tag{19}$$

where the notation $\mapsto$ is used to denote that the new expression is obtained after dropping terms that are seen to contain the $g_{1,0}$ factor. It is also possible to replace $[0, G_2, 0]$ with

$$[0, G_2, 0] \mapsto g_{0,0}\{[G_1, 0] + [0, G_1]\}. \tag{20}$$

Continuing in this manner, one needs to keep from G_3 only the last term and can substitute into it (19) to obtain

$$[0, 0, G_3, 0, 0] \mapsto -g_{0,0}g_{1,1}[0, G_1, 0]. \tag{21}$$

We also deduce from this last expression that

$$g_{3,0} \mapsto -g_{0,0}g_{1,1}g_{1,1}. \tag{22}$$

Finally, the substitution of the terms (19)–(22) into (18) gives

$$\begin{aligned}&[0, 0, 0, G_4, 0, 0, 0] \mapsto \\ &g_{0,0}g_{1,1}(-g_{0,0}g_{1,1})\{\,[0, G_1, 0, 0,] + [0, 0, G_1, 0]\,\} \\ &\quad -(-g_{0,0}g_{1,1}g_{1,1})g_{0,0}\{\,[0, G_1, 0, 0,] + [0, 0, G_1, 0]\,\} = 0.\end{aligned}$$

According to the above $\mapsto$ convention, the last expression says that $G_4(z)$ consists of terms that contain the factor $g_{1,0}$ (and thus were already dropped) plus more terms that sum up to zero. This completes the proof of the first assertion.

To prove the first part of claim (b), substitute into $zF_{n-2}(z)$ the initiation (7) and the implied $f_{n,0} = d_n + d_0$ and $f_{n-1,0} = d_n - d_0$ to obtain

$$zF_{n-2}(z) = \frac{2d_0[zD_n(z) - D_n^\sharp(z)] + 2d_n[D_n(z) - zD_n^\sharp(z)]}{z-1}.$$

For the second part of claim (b), it can be shown that carrying out this division (alone) alters the remainder of the sequence (from the second term and on) in a way that is identical to producing the sequence starting with $F_n(z)$ replaced by $F_n(z)/2$. Then it remains to realize that the proof of part (a) is indifferent to starting it with $G_0 = F_n$ or $G_0 = F_n/2$.

Claim (c), that $F_m(z)/f \in \mathbb{I}[z]$ implies $F_k(z)/f \in \mathbb{I}[z]$ also for $k = m - 1, \ldots, 0$, is obvious from the form of the recursion (8). It implies that the exposed common factors "$g_{1,0}$" propagate and multiply as the recursion proceeds. □

3.2. Fraction-free stability test

Lemma 5 shows that each polynomial $F_m(z)$, $m \leq n-4$ that is produced by Algorithm 2 is divisible by all the previous elements $f_{m+k,0}\, k = 4, \ldots, n-m-1$ and passes them on to future polynomials. The compounded presence of such factors as the recursion goes on explains the rapid growth of integer lengths. Fortunately, Lemma 5 also identifies the buildup mechanism of these factors. The next algorithm implements the insight gained from Lemma 5 and eliminates these common factors in a recursive manner as soon as they are produced.

Algorithm 3: ***Fraction-Free Form.*** *Assume $D_n(z)$ (1) and assign to it a sequence of $n+1$ symmetric polynomials $\{R_m(z) = \sum_{k=0}^{n-m} r_{m,k} z^k,\ m = n, \ldots, 0\}$, as follows:*

$$R_n(z) = D_n(z) + D_n^\sharp(z), \qquad R_{n-1}(z) = \frac{D_n(z) - D_n^\sharp(z)}{(z-1)}. \tag{23}$$

Set $\eta_n = 2$ (or $\eta_n = 1$, see Remark 4), then $\eta_{n-1} = 1$, and for $m = n-1, n-2, \ldots, 1$ do:

$$zR_{m-1}(z) = \frac{r_{m+1,0}(z+1)R_m(z) - r_{m,0}R_{m+1}(z)}{\eta_{m+1}}; \qquad \eta_{m-1} = r_{m,0}. \tag{24}$$

Because Algorithm 3 implements the elimination of factors identified in Lemma 5, it follows that if $D_n(z) \in \mathbb{I}[z]$, then $R_m(z) \in \mathbb{I}[z]$ for all $m = n, \ldots, 0$. The name fraction-free is used to distinguish it from the division-free algorithm and stress that the division it involves is exact, i.e., Algorithm 3 is integer preserving.

Remark 3. A straightforward thinking of how to reduce the coefficients' growth might be carrying a greatest common divisor (gcd) algorithm after each step. This approach might have made sense had our sole purpose been reduction of

common factors. (Though it could be argued that even to such an end it would severely increase the computational complexity.) However, we are interested in systematic elimination of only structured common factors. A brute-force elimination of common factors might also eliminate factors dependent on the tested polynomial. Losing control over the pattern of elimination of factors renders such a technique useless for stability because it becomes impossible to associate to it general necessary and sufficient conditions for stability. This technique can be argued to be equally counterproductive from the perspective of other prospective implementations of the method that this paper proposes.

Normal conditions are defined for Algorithm 3 by

$$r_{m,0} \neq 0 \qquad m = n, n-1, \ldots, 0. \tag{25}$$

Again, they present sufficient conditions for the algorithm to end without interruption and produce a sequence of normal $R_m(z)$ for all $m = n, \ldots, 0$, and, as will be shown in Theorem 7, a broad enough framework for testing stability.

Next we establish relations between the two sequences $\{R_m(z)\}$ and $\{T_m(z)\}$ that will help to associate Algorithm 3 with stability conditions.

Lemma 6. *For any fixed $D_n(z)$, Algorithm 3 is normal if, and only if, Algorithm 1 is normal. Assuming normal conditions, polynomials of respective degrees produced by the two algorithms are proportional. When $\eta_n = 1$, the relations are*

$$R_m(z) = \alpha_m T_m(z), \quad m = n, \ldots, 0 \tag{26}$$

where the scaling factors are given by: $\alpha_n = 1$, $\alpha_{n-1} = 1$ and for $m \leq n-2$

$$\alpha_m = t_{m+1,0}\alpha_{m+1} \quad m \leq n-2 \tag{27}$$

$$\alpha_m = r_{m+1,0} \quad m \leq n-2. \tag{28}$$

For $\eta_n = 2$, denote the sequence produced by $\{\tilde{R}_m(z) = \sum_{k=0}^{n-m} \tilde{r}_{m,k} z^k, \quad m = n, \ldots, 0\}$ (where tildes were added for distinction from the polynomials $R_m(z)$ that, momentarily, will be assumed to correspond only to the choice $\eta_n = 1$). The relations are

$$\tilde{R}_m(z) = \tilde{\alpha}_m T_m(z), \quad m = n, \ldots, 0 \tag{$\tilde{26}$}$$

where for $\tilde{\alpha}_n = \tilde{\alpha}_{n-1} = 1$

$$\tilde{\alpha}_m = h(n-m)\, t_{m+1,0}\tilde{\alpha}_{m+1} \quad m \leq n-2 \tag{$\tilde{27}$}$$

$$\tilde{\alpha}_m = h(n-m)\, \tilde{r}_{m+1,0} \quad m \leq n-2 \tag{$\tilde{28}$}$$

where $h(k) = 1/2$ for even k and $h(k) = 1$ for odd k.

Proof. We shall prove the relations for only $\eta_n = 1$. Note that (28) follows readily from (27). Equation (27) can be proved by induction as follows. Set (26) into the right-hand side of (24) for $m = n-1, n-2$ (note that there is no division yet)

$$\begin{aligned} zR_{m-1}(z) &= r_{m+1,0}(z+1)R_m(z) - r_{m,0}R_{m+1}(z) \\ &= \alpha_{m+1}t_{m+1,0}(z+1)\alpha_m T_m(z) - \alpha_m t_{m,0}\alpha_{m+1}T_{m+1}(z) \\ &= \alpha_{m+1}\alpha_m t_{m,0} z T_{m-1}(z). \end{aligned}$$

Thus $\alpha_{m-1} = \alpha_{m+1}\alpha_m t_{m,0}$ for $m = n-1, n-2$. This verifies (27) for $m = n-2, n-3$ because $\alpha_m = 1$ for $n, n-1$. Now, assuming (for the induction) that (27) holds for $n-2, n-3, \ldots, m$, set (26) into the right hand side of (24) (notice that now the non trivial division is included)

$$\begin{aligned} zR_{m-1}(z) &= \frac{r_{m+1,0}(z+1)R_m(z) - r_{m,0}R_{m+1}(z)}{r_{m+2,0}} \\ &= \frac{\alpha_{m+1}t_{m+1,0}(z+1)\alpha_m T_m(z) - \alpha_m t_{m,0}\alpha_{m+1}T_{m+1}(z)}{\alpha_{m+2}t_{m+2,0}} \\ &= \frac{\alpha_m\alpha_{m+1}t_{m,0}}{\alpha_{m+2}t_{m+2,0}} z T_{m-1}(z) \end{aligned}$$

to obtain

$$\alpha_{m-1} = \frac{\alpha_m\alpha_{m+1}t_{m,0}}{\alpha_{m+2}t_{m+2,0}}.$$

Set into the above $t_{m+2,0} = \alpha_{m+1}/\alpha_{m+2}$, true by the induction assumption, to obtain that (27) holds also for the next step. This completes the proof of (27). □

Remark 4. The choice $\eta_n = 1$ removes the factors depicted in part (a) of Lemma 5. These factors are a feature of the form of the recursion (only). The choice $\eta_n = 2$ adds extraction of powers of 2 depicted by property (b) in Lemma 5 which is dependent also on the choice of the two polynomials that initiate the recursion. The similar proof for the relations ($\tilde{26}$)–($\tilde{28}$) can be drawn except that attention to distinction between steps with $h = 1$ and steps with $h = 1/2$ makes it longer. The use of $\eta_n = 2$ instead of $\eta_n = 1$ leads to elimination of additional common powers of 2 factors up to $2^{\lfloor n/2 \rfloor}$, where $\lfloor n/2 \rfloor$ denotes integer part of $n/2$. Thus the effect of this definite advantage of using $\eta_n = 2$ increases rapidly with the degree of the tested polynomial. The exact pattern of the extra saving is

$$\tilde{R}_{n-2k}(z) = \left(\frac{1}{2}\right)^k R_{n-2k}(z),\ \tilde{R}_{n-2k-1}(z) = \left(\frac{1}{2}\right)^k R_{n-2k-1}(z),\ k = 1, ..., \lfloor n/2 \rfloor. \tag{29}$$

Reference in the following proof to equations (26)–(28) is interchangeable with respect to ($\tilde{26}$)–($\tilde{28}$). All subsequent theorems hold equally for $\eta_n = 2, 1$ and no further distinction between the "tilde" and no "tilde" case will be made. The forthcoming examples use $\eta_n = 2$.

Theorem 7. *A polynomial $D_n(z)$ (1) is stable if and only if $r_{n-1,0}$ $(= d_n - d_0) > 0$ and Algorithm 3 yields for it*

$$R_m(1) > 0 \qquad m = n, n-1, \ldots, 0\ . \tag{30}$$

If $D_n(z)$ is stable, then the following conditions hold as well:

$$r_{m,0} > 0 \qquad m = n, n-1, \ldots, 0. \tag{31}$$

Proof. Assume that Algorithm 3 is normal for $D_n(z)$ and that it produces polynomials such that condition (30) holds and $r_{n-1,0} = d_n - d_0 > 0$. Examine the recursion (24) from the last term upward, after setting into it $z = 1$. $r_{0,0} = R_0(1) > 0$. $r_{1,0} = R_1(1)/2 > 0$. Then, use $r_{3,0}R_0(1) + r_{1,0}R_2(1) = 2r_{2,0}R_1(1)$ and invoke positivity at $z = 1$ of the involved $R_k(z)$ to conclude that $r_{3,0} > 0 \Rightarrow r_{2,0} > 0$. Use similarly $r_{k+1,0}R_{k-2}(1) + r_{k-1,0}R_k(1) = 2r_{k,0}R_{k-1}(1)$ to conclude that $r_{k+1,0} > 0 \Rightarrow r_{k,0} > 0$ also for $k = 3, \ldots, n-2$. The last conclusion $r_{n-1,0} > 0 \Rightarrow r_{n-2,0} > 0$ holds because $r_{n-1,0} > 0$ by assumption. Therefore, $r_{k,0} > 0$ for all $k \leq n-2$. Then (28) implies that all $\alpha_m > 0$. Therefore, (26) implies that all $T_m(1) > 0$ and $D_n(z)$ is stable by Theorem 1.

Conversely, assume that $D_n(z)$ is stable. Then $|d_n| > |d_0|$ (one of a few visible necessary conditions for stability that are often suggested as worth inspection before proceeding with any stability criterion [2, Lemma 2.3]). The assumption $D_n(1) > 1$ in (1) implies $d_n > 0$ (see Remark 1). It follows that $r_{n-1,0} = d_n - d_0 > 0$. Stability implies, according to Theorem 1, that all $T_m(1) > 1$ and all $t_{m,0} > 0$. Therefore, all $\alpha_m > 0$ by (27). Then (26) implies (30). Also, (31) follows via (28). □

Remark 5. Note that the conditions in (30) are not sufficient for stability without $d_n > d_0$. This point is emphasized because it differs from what one might expect after (4) and (13), which present both necessary and sufficient conditions for stability for their respective algorithms. A counter example that illustrates this point will be provided later (see Example 5).

The values $\rho_m := R_m(1)$ required for Theorem 7 may also be calculated with fewer arithmetic operations using the next recursion

$$\rho_{m-2} = \frac{2r_{m,0}\rho_{m-1} - r_{m-1,0}\rho_m}{\eta_m} \tag{32}$$

that has to be initiated by $\rho_n = R_n(1)$, $\rho_{n-1} = R_{n-1}(1)$. It is obtained by setting $z = 1$ into (24) and can be made part of Algorithm 3.

Remark 6. When $D_n(1) < 0$, the following changes in the stability conditions in Theorem 7 can be shown to occur. The necessary and sufficient conditions for stability become $r_{n-1,0} < 0$ and $\mathrm{Sgn}\{R_{n-k}(1)\} = (-1)^k$, $k = 1, \ldots, n$, where Sgn denotes "sign of" (because $R_n(1) = 2D_n(1)$ it may be omitted from the conditions both here and in (30)). The remaining necessary condition in (31) changes to $r_{n,0} < 0$ and $\mathrm{Sgn}\{r_{n-k,0}\} = (-1)^k$, $k = 1, \ldots, n$. It is also possible to arrange the stability conditions for Algorithm 3 here and in Theorem 7 into a combined expression that relates them to Sgn $D_n(1)$ (cf. Theorem 9).

Example 3. We return to the polynomial $D_7(z)$ used to illustrate the division-free

test in Example 1 and apply to it the fraction-free test. Algorithm 2 produces the following stability table for $D_7(z)$.

9	8		9		12		(12)		(9)		(8)		(9)	
	7		9		14		18		(14)		(9)		(7)	
		44		72		102		(102)		(72)		(44)		
			416		602		636		(602)		(416)			
				2120		1720		(1720)		(2120)				
					7300		1880		(7300)					
						16600		(16600)						
							99600							

Clearly, all $R_m(1) > 0$ (the sum of each row is positive) and $r_{n-1,0} = 7 > 0$. Therefore, $D_7(z)$ is stable. Notice the reduction in the length of integers by comparison with Example 1.

The next theorem quantifies the superiority of Algorithm 3 over Algorithm 2 in reducing coefficients' growth. It uses the term coefficient length and the value B as discussed in the paragraph that precedes Theorem 4.

Theorem 8. *The coefficients' length in Algorithm 3 grows linearly with the degree of a $D_n(z) \in \mathbb{I}(z)$. More specifically, assume that B presents the length of $D_n(z)$ (and, as in Theorem 4, also of $R_n(z)$ and $R_{n-1}(z)$); then the length of the coefficients of $R_{n-k}(z)$ is bounded by kB, $k = 2, \ldots, n$.*

Proof. Let $\ell_r(k)$ denote the bound on the length of $R_{n-k}(z) \in \mathbb{I}[z]$. It follows from the algorithm that $\ell_r(k) = \ell_r(k-1) + \ell_r(k-2)$ for $k = 2, 3$ and $\ell_r(k) = \ell_r(k-1) + \ell_r(k-2) - \ell_r(k-3)$ for $k \geq 4$. By assumption, $\ell_r(0) = B$ and $\ell_r(1) = B$. Therefore, the second difference equation has to be solved for the initial values $\ell_r(1) = B$, $\ell_r(2) = 2B$, $\ell_r(3) = 3B$. The solution is $\ell(k) = kB$ for all $k \geq 1$. In particular, the maximal length for the last entry of the table is nB. □

Higher-degree polynomials, which may be encountered in practice, will exhibit a more dramatic reduction in the coefficients' length than the comparison of Example 3 to Example 1 reveals.

Example 4. Here we try again to find the stability intervals for K in the polynomial $D_7(z, K)$ considered in Example 2. The properties shown for Algorithm 2 imply that all the coefficients of the polynomials are now in $I[k]$ and not only the integers but also the powers of K grow linearly from step to step. The algorithm is initiated with the pair of polynomials:

$$R_7(z) = 8 + K + 8z + 9z^2 + 12z^3 + 12z^4 + 9z^5 + 8z^6 + 8z^7 + Kz^7$$
$$R_6(z) = 8 - K + (10 - K)z + (15 - K)z^2 + (19 - K)z^3 + (15 - K)z^4 + (10 - K)z^5 + (8 - K)z^6.$$

After completing all the remaining polynomials (we shall skip them), one has to set into them the value $z = 1$ to obtain

$$R_7(1) = 74 + 2K$$

$$R_6(1) = 85 - 7K$$
$$R_5(1) = 384 + 58K - 6K^2$$
$$R_4(1) = 2744 + 15K - 92K^2 + 5K^3$$
$$R_3(1) = 6704 + 1382K - 356K^2 - 54K^3 + 4K^4$$
$$R_2(1) = 22424 - 3309 * K - 2792K^2 + 104K^3 + 56K^4 - 3K^5$$
$$R_1(1) = 49760 - 2836K - 12204K^2 - 1778K^3 + 230K^4 + 30K^5 - 2K^6$$
$$R_0(1) = \frac{(7-K)R_1(1)}{2}$$

The last expression incorporates a useful simplification that has a general validity. Namely, $R_1(1)/2 \in \mathbb{I}$ and it is a factor of $R_0(1)$. To prove this property, note that $R_1(1) = 2r_{1,0}$ and $R_0(z) = r_{1,0}(2r_{2,0} - r_{2,1})/r_{3,0}$. Next the stability conditions in Theorem 7 have to be examined. The first three inequalities, $R_7(1) > 0$, $R_6(1) > 0$, $R_5(1) > 0$, hold commonly for only $-4.799 < K < 7.104$. The latter interval already also fulfills the condition $r_{1,0} = 8 - K > 0$. The subsequent three requirements, $R_4(1) > 0$, $R_3(1) > 0$, and $R_2(1) > 0$, keep shrinking the admissible interval until $-3.938 < K < 2.517$. Next $R_1(1) > 0$ reduces it to $-3.812 < K < 1.758$, and finally $R_0(1) = (7-K)R_1(1)/2 > 0$ poses no further restriction. Thus stability holds for $-3.812 < K < 1.758$, and (because we tested conditions that are necessary and sufficient for stability) this is also the largest interval of stability for K.

Recall that trying to solve this problem in Example 3 led to polynomial inequalities with degrees in K up to 21. Solving the above problem with the test in its original form would lead to a set of inequalities posed on rational functions so that pairs of numerator and denominator polynomials in K would have to be checked to be either both positive or both negative.

The next example is designed to prove that (30) are not sufficient conditions for stability (cf. Remark 5).

Example 5. Consider $D_4(z) = 3 - 33z + 84z^2 - 24z^3 + z^4$. The resulting fraction-free stability table is

4		−57		168		(−57)		(4)
	−2		7		(7)		(−2)	
		−47		196		(−47)		
			31		(31)			
				4495				

The sum of rows $\{R_4(1), \ldots, R_0(1)\} = \{62,\ 10, 102\ , 62\ , 4495\}$ are all positive, thus the conditions (30) hold. However, $r_{n-1,0} = -2 \not> 0$. Therefore, according to Theorem 7, the polynomial is not stable. (We shall return to this polynomial in Example 6.)

It is beyond the scope of this paper to generalize the proposed stability test into a general IP zero location method. However, the framework of the current

exposition is broad enough to readily generalize the test here to also determine the zero location of a polynomial for which Algorithm 3 remains normal. In the following, a zero z_i of $D(z)$, $D(z_i) = 0$ is called an inside unit circle (IUC) or unit circle (UC) or outside unit circle (OUC) zero if $|z_i| < 1$, or $|z_i| = 1$, or $|z_i| > 1$, respectively.

Theorem 9. *Assume that Algorithm 3 produces for $D_n(z)$ for which $D_n(1) \neq 0$ (the sign may be either positive or negative) a sequence $\{R_m(z), \quad m = n, \ldots, 0\}$ of normal polynomials (i.e., (25) holds). Then $D_n(z)$ has ν OUC zeros and $n - \nu$ IUC zeros (and no UC zeros), where ν presents the number of sign variations in the sequence*

$$\nu = \mathrm{Var}\{\gamma_n, \ldots, \gamma_0\}, \tag{33}$$

where $\gamma_m := R_m(1)\eta_m$.

Proof. The proof of the theorem follows, using relations established in Lemma 6, from the extension of Theorem 1 (in this paper) to a zero location rule in [2, Theorem 2.2]. Accordingly, assuming that Algorithm 1 obeys normal conditions (6), then $D_n(z)$ has $\nu = \mathrm{Var}\{T_n(1), \ldots, T_0(1)\}$ OUC and $n - \nu$ IUC zeros. □

It is notable that the stability conditions in Theorem 7 are stated in a more refined manner than what setting $\nu = 0$ in Theorem 9 immediately implies.

Example 6. Let us return to the polynomial $D_4(z)$ in Example 5 to illustrate Theorem 9. Taking from there $\{R_4(1), \ldots, R_0(1)\} = \{62,\ 10,\ 102\ ,\ -94\ ,\ 44495\}$ and $\{\eta_4, \ldots, \eta_0\} = \{1,\ 1,\ -2\ ,\ -94\ ,\ 62\}$ and setting them into (33) gives

$$\nu = \mathrm{Var}\{62 \cdot 1,\ 10 \cdot 1,\ 102 \cdot (-2), 4495 \cdot (-94),\ 44495 \cdot 62\} = 2.$$

Therefore, according to Theorem 9, $D_4(z)$ has $\nu = 2$ OUC zeros and $4 - \nu = 2$ IUC zeros.

4. Complexity

An adequate measure to evaluate the efficiency of an IP algorithm is counting the number of binary operations it requires, which also amounts to the "computing time" [1]. Let ADD and MULT refer to addition and multiplication operations between elements that correspond to the way we measured the length of coefficients. If, say, in Theorems 4 and 8, lengths B and so forth are measured in numbers of bits, then ADD presents addition (or substraction) and MULT multiplications (or exact divisions) between bits. We assume the case of the normal algorithm. The other assumptions that underlie the following estimated counts are as follows. (i) The sum and difference of two integers of lengths b_1 and b_2 require $\max(b_1, b_2) + 1$ ADDs. (ii) Their product costs $(b_1 + 1)(b_2 + 1)$ FLOPs (i.e., MULT+ADDs). (iii) The exact division of an integer of length $b_1 + b_2$ by a divisor of length b_2 yields an integer of length b_1 and costs $b_2(b_2 + 1)/2$ FLOPs (assuming, as the case is here, that $b_1 < b_2$).

Theorem 10. *The integer computational complexity (or "binary computational cost" or "computing time") of the fraction-free test (Algorithm 3) is given by* $\frac{5}{48}B^2n^4$ *+ terms with lower powers of B and/or n FLOPs.*

Proof. These assumptions on counting binary operations make the complexity evaluation comparable to counting real arithmetic operations for an integral ring of polynomials. Therefore, the cost evaluation in [6, Section 5.2] can be used. The figure $\frac{5}{2}n_1^2n_2^4$ in [6] should be translated to our need here through the correspondences: $n_1 \leftrightarrow B$ (the measures for the two initial conditions) and $n_2 \leftrightarrow n/2$ (the 1/2 appears because a doubling of degree that is used in [6] is not relevant for the setting here). The figure in the theorem follows at once. □

It can be similarly deduced from analogy with the evaluation of the intermediate algorithm in [6, Section 3.4] that the division-free test, i.e., Algorithm 3 has an exponential complexity dominated by the term $B^2\lambda_o^{2n}$, where $\lambda_o := \frac{1+\sqrt{5}}{2} > 1$ (cf. Theorem 4).

The MJT stability test [10], [11], [12], which is the only other available IP 1D stability test [1], has binary complexity with a leading term of $\frac{5}{2}B^2n^4$ FLOPs. This figure can be deduced from the cost analysis held in [4] for the 2D stability testing table based on the MJT test in [9]. Accordingly, the current IP stability test is more efficient binary-wise (is faster in computing time) by approximately a factor of 24.

Obviously, the use of the proposed stability test need not be restricted to integer polynomials. In terms of conventional count of real arithmetic operations, the complexity counts look different. The count of real operations is higher for an IP property than for the original form of the stability test. The original test requires approximately $\frac{1}{4}n^2$ multiplications and $\frac{1}{2}n^2$ additions. The division-free test has two multipliers in the recursion and hence requires $\frac{1}{2}n^2$ multiplications and $\frac{1}{2}n^2$ additions. The latter would also be the cost of the fraction-free test if it is carried out with two multipliers ($\frac{r_{m+1,0}}{\eta_{m+1}}$ and $\frac{r_{m,0}}{\eta_{m+1}}$). However, the preferable procedure (even for real polynomials) is to first compute the numerator on the right-hand side of (24) and only afterward divide it. This mode of calculation adds to the overall count $\frac{1}{4}n^2$ divisions. However, it is expected to improve numerical accuracy when testing a real polynomial because the algorithm then works in a manner that collaborates with the way real numbers are registered and manipulated by a compute (because binary mathematics resembles operation over a power-of-two ring of polynomials). Of course, when using the stability test to real polynomials given by decimal coefficients, it is possible to enforce increased accuracy simply by turning the tested polynomial into an integer polynomial by scaling it up properly.

5. Conclusion

This paper has presented a modification of the author's stability test [2], [5], which turns it into an efficient IP stability test for discrete-time linear systems.

The paper illustrated the possible advantage of the new test as a design tool with symbolic computational examples and illuminated its capacity to increase numerical accuracy when testing ill-conditioned or high-degree real polynomials.

The proposed test may also be used over other integer-like domains (integral rings). Successful demonstrations of an implementation within this capacity are the stability tests for 2D discrete-time systems in [6], [7]. The test may also affect digital signal processing and filter design algorithms related to stability testing algorithms.

References

[1] P. G. Anderson, M. R. Garey, and L. E. Heindel, Combinational aspects of deciding if all roots of a polynomial lie within the unit circle, *Computing*, vol. 16, pp. 293–304, 1976.

[2] Y. Bistritz, Zero location with respect to the unit circle of discrete-time linear system polynomials, *Proc. IEEE,* vol. 72, pp. 1131–1142, Sept. 1984.

[3] Y. Bistritz, Reflection on Schur–Cohn matrices and Jury–Marden tables and classification of related unit circle zero location criteria, *Circuits Systems Signal Process.*, vol. 15, no. 1, pp. 111–136, 1996.

[4] Y. Bistritz, Stability testing of two-dimensional discrete-time systems by a scattering-type tabular form and its telepolation, *Multidimen. Systems Signal Process.*, vol. 13, pp. 55–77, Jan. 2002.

[5] Y. Bistritz, Zero location of polynomials with respect to the unit-circle unhampered by nonessential singularities, *IEEE Trans. Circuits and Systems Part I*, vol. 49, pp. 305–314, Mar. 2002.

[6] Y. Bistritz, Real-polynomial based immittance-type tabular stability test for two-dimensional discrete systems, *Circuits Systems Signal Process.*, vol. 22, no. 3, pp. 255–276, 2003.

[7] Y. Bistritz, Testing stability of 2-D discrete systems by a set of real 1-D stability tests, *IEEE Trans. Circuits and Systems Part I*, to appear.

[8] Y. Bistritz, H. Lev-Ari, and T. Kailath, Immittance domain three-term Levinson and Schur recursions for quasi-Toeplitz complex Hermitian matrices, *SIAM J. Matrix Anal. Appl.*, vol. 12, pp. 497–520, July 1991.

[9] X. Hu and E. I. Jury, On two-dimensional filter stability test, *IEEE Trans. Circuits and Systems,* Vol. 41, pp. 457–462, July 1994.

[10] E. I. Jury, A modified stability table for linear discrete systems, *Proc. IEEE*, vol. 53, pp. 184–185, Feb. 1965.

[11] E. I. Jury, Modified stability table for 2-D digital filter, *IEEE Trans. Circuits and Systems*, vol. 35, pp. 116–119, Jan. 1988.

[12] E. I. Jury, A note on the modified stability table for linear discrete time system, *IEEE Trans. Circuits and Systems*, vol. 38, 221–223, 1991.

[13] S. J. Orfanidis, *Optimum Signal Processing: An Introduction*, 2nd edition, Macmilan, New York, 1988.