Optimal Fraction-Free Routh Tests for Complex and Real Integer Polynomials

Yuval Bistritz, *Fellow, IEEE*

***Abstract*—The Routh test is the simplest and most efficient algorithm to determine whether all the zeros of a polynomial have negative real parts. However, the test involves divisions that may decrease its numerical accuracy and are a drawback in its use for various generalized applications. The paper presents fraction-free forms for this classical test that enhance it with the property that the testing of a polynomial with Gaussian or real integer coefficients can be completed over the respective ring of integers. Two types of algorithms are considered one, named the G-sequence, which is most efficient (as an integer algorithm) for Gaussian integers, and another, named the R-sequence, which is most efficient for real integers. The G-sequence can be used also for the real case, but the R-sequence is by far more efficient for real integer polynomials. The count of zeros with positive real parts for normal polynomials is also presented for each algorithm.**

***Index Terms*—Continuous-time systems, integer algorithms, linear network analysis, polynomials, Routh-Hurwitz criterion, stability.**

I. Introduction

ROUTH devised in 1877 [1] an algorithm to determine whether a real polynomial $P(s) = \sum_{k=0}^{n} p_k s^k$ has all its zeros in the left half-plane, $\mathcal{R}e\{s\} < 0$. We shall refer to a polynomial with this property as 'stable' (the term Hurwitz polynomial is also widely used) because a continuous-time constant linear system with a stable characteristic polynomial is stable. The Routh test involves just $0.25n^2 + O(n)$ of standard arithmetic operations making it the most efficient algorithms to test stability of $P(s)$. It is worth noting that Routh was motivated by a practical stability problem that was proposed to him by Maxwell and he won a respectable prize for his solution.

The paper considers integer preserving (IP) Routh tests for complex and real integer polynomials. The IP property means that when the test is applied to a polynomial whose coefficients are Gaussian integers or real integers, all the calculations can be completed over the respective integral domain (i.e., without encountering rational numbers). The algorithms are called fraction-free (FF) to emphasize that they remain over the integers even though they do involve exact divisions (cancellation of common factors of the coefficients). The adjective optimal is a saying about the efficiency of the algorithms as integer algorithms that will be clarified later. The FF property makes the test a more suitable system design tool with nowadays symbolic language softwares. It also guarantees accuracy for high degree and ill conditioned polynomials because integer arithmetics is exact. More prospective applications will arise through comments and citations along the paper. Therefore, the discussion of usages for FF Routh tests will be resumed in the concluding remarks.

The problem is often called the Routh-Hurwitz problem because Hurwitz, again motivated by a stability problem, derived in [2] (18 years after Routh but independently) a stability criterion that requires the positivity of all the principal minors of an $n \times n$ (not symmetric) matrix (known today as the Hurwitz matrix) whose entries are a simple outlay of the coefficients of the real polynomial. The Hurwitz criterion provides important supporting theory, but testing stability by direct evaluation of the sequence of Hurwitz determinants (without attending to their relation to the Routh test) has higher order of complexity than the Routh test. The method belongs to a group of stability criteria, often called determinant methods, that express stability by conditions posed on matrices. These alternative methods started earlier with works by Bezout, Sylvester and Hermite (who, lacking the stability incentive, actually considered criteria for the closely related problem of zeros in the upper half-plane). Since then, matrix stability methods have been the subject of a vast amount of publications. Due to space limitation, we shall keep out of the current scope matricial aspects of the theory. A gentle introduction to both the Routh test and related determinant methods with rich bibliographical notes can be found in Barnett's book [3]. The earlier books of Marden and Gantmacher also devote special chapters to the Routh-Hurwitz problem [4], [5] .

The Routh test has been presented in the literature in several ways. Originally, Routh presented it in an array that has become known as the Routh table. This is still the popular way for presenting the test in undergraduate textbooks on linear systems. Continued fraction expansions (CFE) were used by Wall [6] and by Frank who, advised by Wall, generalized the Routh algorithm to complex polynomials in [7]. The CFE presentation attracted analog circuit theory researchers like Cauer, Darlington and others as a means to realize two-element-kind networks by RLC ladders, see e.g., [8]. A third possible presentation for the Routh test is by three-term recursions of polynomials. This will be the form used also in the current paper because it is compact and very convenient for manipulations and proofs.

Much attention in the literature on the Routh test was directed to ways to overcome certain singular cases. Singular cases must be resolved in order to generalize the stability test into a full zero location (ZL) procedure, one that can determine the distribution of zeros with respect to the imaginary axis for *any* polynomial. The current paper restricts the presentation to the non-singular case (which is broad enough for testing stability). However, as long as the recursion remains non-singular, the paper goes beyond stability and presents for each algorithm also ZL rules. A

Manuscript received June 21, 2012; revised October 19, 2012; accepted January 07, 2013. This paper was recommended by Associate Editor M. Frasca.

Y. Bistritz is with the School of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel (e-mail: bistritz@eng.tau.ac.il).

Digital Object Identifier 10.1109/TCSI.2013.2246232

comprehensive treatment of singularities in the complex Routh test appeared in [9].

Some related research considered the positioning of the original Routh test in a larger family of polynomial recursions that can equally perform the same task. The original Routh test uses a three-term polynomial recursion with degree reduction achieved by extraction at $s = \infty$ (elimination of the leading coefficients of successive pairs of polynomials). However, it is possible to devise variants that use other points of extraction. It is also possible to use, instead of three-term polynomial recursions, two-term polynomial recursions reminiscent of those used in the classical stability tests for discrete systems by Schur Cohn Marden and Jury (in the following 'the SCMJ class of tests' [10]). Research on these issues includes [11], [12] and [13] (for real polynomials) and [14] (that considers also complex polynomials). It follows that two-term polynomial recursion versions are less efficient algorithms with less attractive settings and zero location rules than those with three-term recursions. Among the three-term recursion tests, the only equally simple alternative to the original setting occurs when the extraction point is at $s = 0$ (that will be used in this paper). So these studies, beyond their theoretical value, reinforce the success of Routh to hit from the start a stability test that has never been surpassed in efficiency and simplicity since its introduction in 1877.

The paper presents FF Routh tests that involve three-term recursions in two, not trivially related, forms. One is designed for polynomials with Gaussian (i.e., complex) integer coefficients and the other is specific for real integer polynomials. The main result for complex polynomials is named the G-sequence and is stated in Algorithm 2. The other main result for the real case is named the R-sequence and is stated in Algorithm 5. Each algorithm is shown to be the most efficient integer algorithm for its designed task. By comparison with the G-sequence, that is of course a valid integer algorithm also for the real case, the R-sequence produces integers of half size and is by far a more efficient integer algorithm for a real polynomial (where the terms 'size of integers' and 'efficiency of integer algorithms' will be clarified later).

The research reported in this paper stems from seeking Routh analogs for FF versions of the Bistritz test that were discovered during its application to testing stability of two-dimensional (2-D) discrete systems, see the complex "accompanying 1-D stability test" in [15] and the real test in [16] (and references there in). The original (not FF) Bistritz test is a Routh-like stability test for discrete-time systems introduced about three decades ago [18] (this later reference is proposed instead because it brings the test in its most accomplished form as a general unit-circle ZL procedure). Unlike the classical unit-circle stability tests (the aforementioned SCMJ class) that use two-term recursions to propagate polynomials of no specific structure, the newer test showed that it is possible and more efficient (in counts of standard arithmetic operations) to use a certain three-term recursion that propagates symmetric polynomials. The Bistritz test also evoked the above mentioned studies on alternative two-term and three-term Routh tests [11]–[14] and inspired studies of alternative algorithms in some more contexts (e.g., the Levinson algorithm) called collectively 'immittance' (or 'split') algorithms for distinction from their classical 'scattering' counterparts.

One can not be too careful in claiming a new contribution in a field with a long history like the Routh-Hurwitz problem. While scanning the literature to wrap up this paper, we discovered that our R-sequence algorithm is similar to the 'optimal' Routh array proposed by Jeltsch [19]. The other contributions in this paper seem to be fully new. This includes the complex FF Routh test (the G-sequence), some intermediate algorithms, the relation among the complex and the real FF algorithms, and the accompanying stability and zero location rules. By closer comparison of the R-sequence with the Routh array in [19], the R-sequence here is presented by a polynomial recursion (and a reversed setting), the FF property is proved differently, the algorithm is accompanied with stability and zero location rules, and it is put in context and relationship with the complex FF case. As a matter of fact, the paper establishes the G-sequence and the R-sequence and their properties in a fully self contained manner, without reliance on extraneous stability criteria or matrix manipulations. Jeltsch used the term optimal to describe the minimal size of integers that the array attains. We adopted this term for the title of our paper and shall argue that both the G-sequence and the R-sequence are most efficient stability tests—one for a general Gaussian integer and the other for real integer polynomial, where the efficiency of the integer algorithms is measured by their *binary complexity*, see e.g., [20].

The paper is constructed as follows. The next section presents the complex Routh test that will serve as a template for further evaluations. It differs somewhat from the classical setting in ways that will be discussed. Section III is devoted to FF Routh tests for Gaussian integer polynomials. Section IV considers real integer Routh tests. Section V compares the efficiency of all the presented FF Routh tests as integer algorithms and the paper concludes with some remarks.

II. The Routh Test for Complex Polynomials

Let $\mathbb{R}$ and $\mathbb{C}$ denote the fields of real and complex numbers, respectively. The test is assumed to be applied to a polynomial $P(s) \in \mathbb{C}[s]$ of the form

$$P(s) = \sum_{k=0}^{n} p_k s^k = \sum_{k=0}^{n} (p'_k + \mathrm{j}p''_k)s^k,\ p_n p'_0 \neq 0 \qquad (1)$$

where $p'_k, p''_k \in \mathbb{R}$ and $\mathrm{j} = \sqrt{-1}$. Let $\bar{P}(s) = \sum_{k=0}^{n}(p'_k - \mathrm{j}p''_k)s^k$ denote its conjugated-coefficient polynomial and $\tilde{P}(-s) = \sum_{k=0}^{n}(p'_k - \mathrm{j}p''_k)(-s)^k$ its para-conjugate polynomial. The polynomial can be decomposed into its even and odd parts,

$$P(s) = P_o(s) + P_e(s) \qquad (2)$$

$$P_o(s) = \frac{P(s) - \bar{P}(-s)}{2} = \mathrm{j}p''_0 + p'_1 s + \mathrm{j}p''_2 s^2 + p'_3 s^3 + \cdots$$
$$P_e(s) = \frac{P(s) + \bar{P}(-s)}{2} = p'_0 + \mathrm{j}p''_1 s + p'_2 s^2 + \mathrm{j}p''_3 s^3 + \cdots \qquad (3)$$

where a polynomial is called even if $\bar{P}_e(-s) = P_e(s)$, and odd if $\bar{P}_o(-s) = -P_o(s)$. Even and odd complex polynomials ex-

hibit alternatingly real and pure imaginary coefficients as illustrated above.

Our reference for all the forthcoming FF algorithms will be the next complex Routh algorithm.

Algorithm 1: The D-sequence Construct for a polynomial $P(s) \in \mathbb{C}[s]$ as in (1) a sequence of polynomials $\{D_m(s) = \sum_{k=0}^{m} d_{m,k}s^k, m = n, n-1, \ldots, 0\}$ as follows.

Initiation: Set $A_n(s) = \sum_{k=0}^{n} a_{n,k}s^k = P_o(s)$ and $D_n(s) = P_e(s)$ to the even and odd parts given in (3). Obtain $D_{n-1}(s)$:

$$sD_{n-1}(s) = -\alpha_n D_n(s) + A_n(s)\ ,\ \alpha_n = \frac{a_{n,0}}{d_{n,0}}. \tag{4}$$

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

$$\begin{aligned} \beta_m =& \frac{d_{m+1,0}}{d_{m,0}} \\ \alpha_m =& \frac{d_{m+1,1} - \beta_m d_{m,1}}{d_{m,0}} \end{aligned} \tag{5}$$

$$s^2 D_{m-1}(s) = -(\alpha_m s + \beta_m)D_m(s) + D_{m+1}(s). \tag{6}$$

Note that the completion of the algorithm requires that $D_m(0) \neq 0$ for $m = n-1, n-2, \ldots, 1$. We add to this $D_n(0) = p'_0 \neq 0$ (assumed already in (1) with no loss of generality) and $D_0(0) \neq 0$ and say that the algorithm obeys normal conditions for $P(s)$ if

$$D_m(0) \neq 0, \quad m = n, n-1, \ldots, 0. \tag{7}$$

It follows from the next theorem that normal conditions are necessary conditions for stability. The presentation in this paper will be restricted throughout to such normal conditions. However, given normal conditions, we shall always bring not just stability conditions but also ZL rules.

Theorem 1: Assume that Algorithm 1 produces for $P(s) \in \mathbb{C}[s]$ as in (1) a normal D-sequence. Then $P(s)$ has ν zeros in the right half-plane (RHP), $\mathcal{R}e\{s\} > 0$, and $n - \nu$ zeros in the left half-plane (LHP), where ν is the number of sign variations of the sequence at $s = 0$, viz.

$$\nu = Var\{D_n(0), D_{n-1}(0), \ldots, D_1(0), D_0(0)\}. \tag{8}$$

Usually we shall write the last term in a sign variation sequence as above (i.e., $D_0(0)$), even though the last polynomial of the sequence will always have zero degree (e.g., here $D_0(s) = d_{0,0}$). Note that ν is equivalently given by the number of negative β_m's

$$\nu = n_-\{\beta_{n-1}, \ldots, \beta_0\}. \tag{9}$$

A proof of Theorem 1 is brought in Appendix A. It uses the so called argument principle (or Rouché's theorem) which is also the tool used by Frank [7]. Spelling out a proof seems to be the best way to cover a couple of diversions of the D-sequence here from the form it assumed by Frank that will be discussed in the last paragraph of this section. In addition, since all forthcoming ZL rules will be deduced from Theorem 1, bringing for it a rigorous full proof supports our effort to make the derivation of the FF Routh tests in this paper fully self contained.

Remark 1: Note that Theorem 1 implicitly states that there are no zeros on the imaginary axis $\mathcal{I} = \{s|\mathcal{R}e\{s\} = 0\}$. This happens as a direct consequence of the GCD (greatest common divisor) property of the recursion. Let us briefly discuss this property because it will be used repeatedly as an argument in several forthcoming proofs. The recursion (6) can be regarded as an Euclidean algorithm. So it acts as a GCD algorithm on its two initial polynomials (as well as on any subsequent successive pair of polynomials in the sequence). It then becomes evident that it also acts as a GCD algorithm on $P(s)$ and $\bar{P}(-s)$. Suppose now that the algorithm produces a sequence such that a $D_k(s)$, $k > 0$, is followed by $D_{k-1}(s) \equiv 0$. Then $D_k(z)$ contains all the common zeros of $P(s)$ and $\bar{P}(-s)$ (that necessarily are either zeros on the imaginary axis, or occur in pairs $s_i, -\bar{s}_i$). Since normal conditions imply termination with $\gcd\{P(s), \bar{P}(-s)\} = d_{0,0} \neq 0$. It follows that normal conditions imply no zeros on $\mathcal{I}$.

The case $\nu = 0$ presents necessary and sufficient conditions for stability. Notice that normal conditions are necessary for stability and thus provide a broad enough framework for testing it. A practical use of this observation is that it is possible to conclude that a polynomial is not stable (has at least one zero in $\mathcal{R}e\{s\} \geq 0$) as soon as a $D_m(0) = 0$ is encountered (a situation known as 'singularity'). On the other hand, when the sequence ends normally, the rule (8) provides full information on the location of the zeros with respect to the imaginary axis. This will be the situation also in all the forthcoming algorithms. Assuming normal conditions, each algorithm will be associated with an expression for $0 \leq \nu \leq n$ such that the polynomial has ν RHP and $n - \nu$ LHP zeros (and no zeros on $\mathcal{I}$).

Remark 2: The polynomials $\{D_m(s), m = n, n-1, \ldots, 0\}$ are all even, $\bar{D}_m(-s) = D_m(s)$. Consequently, $d_{m,2i} \in \mathbb{R}$ and $d_{m,2i+1} \in \mathcal{I}$ alternatingly, and the free terms $d_{m,0} = D_m(0)$ is real (and also nonzero by the assumption of normal conditions). It follows that $\beta_m \in \mathbb{R}$ and nonzero and that $\alpha_m \in \mathcal{I}$ (possibly zero). Incidently notice that it also follows that using a simple substitution, $s = \mathrm{j}\omega$, the algorithm could be arranged into a recursion of real polynomials in the argument ω and be carried out with only real arithmetics.

Example 1: To illustrate the D-sequence and forthcoming Gaussian integer Routh algorithm we shall use the polynomial:

$$\begin{aligned} P^{(1)}(s) =& -7 - 5\mathrm{j} - (2+\mathrm{j})s + (3+\mathrm{j})s^2 + (3-2\mathrm{j})s^3 \\ &+ (4+2\mathrm{j})s^4 + (3-2\mathrm{j})s^5. \end{aligned} \tag{10}$$

Applying to it Algorithm 1 produces

$$A_5(s) = -5\mathrm{j} - 2s + \mathrm{j}s^2 + 3s^3 + 2\mathrm{j}s^4 + 3s^5$$
$$D_5(s) = -7 - \mathrm{j}s + 3s^2 - 2\mathrm{j}s^3 + 4s^4 - 2\mathrm{j}s^5$$
$$\alpha_5 = 0.7143\mathrm{j}$$
$$D_4(s) = -2.7143 - 1.1429\mathrm{j}s + 1.5714s^2 - 0.85714\mathrm{j}s^3 + 1.5714s^4$$
$$\beta_4 = 2.5789,\ \alpha_4 = -0.71745\mathrm{j}$$
$$D_3(s) = -0.23269 + 1.3379\mathrm{j}s + 0.56233s^2 - 0.87258\mathrm{j}s^3$$
$$\beta_3 = 11.665,\ \alpha_3 = 71.985\mathrm{j}$$
$$D_2(s) = 91.324 - 31.158\mathrm{j}s - 61.241s^2$$
$$\beta_2 = -0.00255,\ \alpha_2 = 0.0138\mathrm{j}$$

$D_1(s) = -0.02310 - 0.02850\mathrm{j}s$
$\beta_1 = -3953.04,\ \alpha_1 = 6242.23\mathrm{j}$
$D_0(s) = -239.761.$

Use (8) or (9) to obtain $\nu = 2$. Therefore, the polynomial has 2 RHP zeros and 3 LHP zeros.

Notice that for a $P(s) \in \mathbb{C}[s]$ with $p_0 \in \mathbb{R}$, the recursion can be initiated with the pair of polynomials:

$$\begin{array}{rcl} D_n(s) = P_e(s) & = & p_0 + \mathrm{j}p_1''s + p_2's^2 + \cdots \\ D_{n-1}(s) = \frac{P_o(s)}{s} & = & p_1' + \mathrm{j}p_2''s + p_3's^2 + \cdots \end{array}. \quad (11)$$

When $P(s) \in \mathbb{R}[s]$, all $\alpha_m = 0$ and the polynomials $D_m(s)$ become real and even (have only powers of s^2). The algorithm reduces to the classical Routh test (up to reversion). It starts with the pair (11) and proceeds for $m = n-1, \ldots, 1$ with the recursion

$$\beta_m = \frac{d_{m+1,0}}{d_{m,0}},\ s^2 D_{m-1}(s) = -\beta_m D_m(s) + D_{m+1}(s). \quad (12)$$

The ZL rules is still given of course by Theorem 1.

Example 2: We shall use the next polynomial to illustrate algorithms for real polynomials.

$$P^{(2)}(s) = 30 + 71s + 84s^2 + 66s^3 + 31s^4 + 10s^5 + 2s^6. \quad (13)$$

The D-sequence can be initiated with (11)

$$\begin{array}{rcl} D_6(s) & = & 30 + 84s^2 + 31s^4 + 2s^6 \\ D_5(s) & = & 71 + 66s^2 + 10s^4 \end{array} \quad (14)$$

and the rest of it obtained by the recursion spelled out for the real case in (12),

$$\begin{array}{ll} \beta_5 = 0.42253\,, & D_4(s) = 56.1127 + 26.7746s^2 + 2s^4 \\ \beta_4 = 1.26531\,, & D_3(s) = 32.1217 + 7.46938s^2 \\ \beta_3 = 1.74688\,, & D_2(s) = 13.7266 + 2s^2 \\ \beta_2 = 2.34011\,, & D_1(s) = 2.78915 \\ \beta_1 = 4.92142\,, & D_0(s) = 2. \end{array}$$

By Theorem 1, $P^{(2)}(s)$ is stable.

Algorithm 1 differs somewhat from the Routh test for complex polynomials of Frank [7]. Frank, uses (a CFE that is equivalent to) a three-term recursion with extraction at $s = \infty$ which has been the customary form in the literature also for the more familiar real Routh test. He also assumes (with no loss of generality for his study) that the given polynomial, say $Q(s) = \sum_{k=0}^{n} q_k s^k \in \mathbb{C}(s)$ is monic, $q_n = 1$. If n is even (resp. odd), then the first polynomials is the odd (resp. even) and the second is the even (resp. odd) parts of $Q(s)$. If one assumes a $P(s) \in \mathbb{C}[s]$ with $p_0 = 1$ and take $Q(s)$ to be the reversion of $P(s)$, i.e., $Q(s) = \sum_{k=0}^{n} p_{n-k} s^k$, than Frank's algorithm would produce a sequence of polynomials related by reversion to the polynomials of the D-sequence. (Note that $P(s)$ and $Q(s)$ have the same distribution of zeros with respect to the imaginary axis.) The D-sequence uses instead a three-term recursion with extraction at $s = 0$. It also removes the restriction $p_0 = 1$ because the assumption $p_0 = 1$ (and even $p_0 \in \mathbb{R}$) is restrictive for the forthcoming derivation of a general Gaussian integer algorithm. One consequence is that the D-sequence has a same initiation for both parities of n, a nicety that is maintained also in the real case and in all the forthcoming FF tests. A second difference is that all the polynomials of the sequence (here as well as in forthcoming algorithms) are even (in the classical setting they are alternatingly even and odd). The third difference is that in the current setting the stability and the ZL rule are posed (here and in all subsequent algorithms) on the free coefficient of the polynomials instead of on the leading coefficient of the polynomials in the classical setting. Note that removing the restriction $p_0 \in \mathbb{R}$ on a $P(s) \in \mathbb{C}[s]$ introduces an extra preliminary step (4) (that is not needed (11) if $p_0 \in \mathbb{R}$). From here on, we adhere to just our chosen setting. However, should there be a reason, this and any of the forthcoming FF Routh tests can be readily converted into the dual setting by applying the reversion operation.

III. FF Tests for a Complex Polynomial

Let $\mathbb{I}$ denote the ring of real integers, and $\mathbb{I}_\mathrm{j}$ the ring of Gaussian integers defined by $\mathbb{I}_\mathrm{j} = \{a + \mathrm{j}b | a, b \in \mathbb{I}\}$. We want to test the stability of a complex polynomial $P(s)$ by an algorithm that is IP. Namely, we want an algorithm that has the property that if $P(s) \in \mathbb{I}_\mathrm{j}[s]$ then it can be fully carried out by operations within $\mathbb{I}_\mathrm{j}$ (i.e., without encountering rational numbers).[1] As Example 1 demonstrates, the D-sequence is not IP. An obvious way to turn it into an integers algorithm is to avoid divisions. We shall see in a moment that this simple approach creates an integer algorithm that is quite not attractive, though it will serve as a good beginning for further improvements. Before that, we must say some words on size of integers and efficiency of integer algorithms.

The efficiency of an algorithm carried out over integers is not measured well by the conventional count of arithmetic operations because a simple count ignores the fact that multiplication of big integers is more expensive than that of small integers. The efficiency of an integer algorithm is measured instead by what is called *binary complexity* [20] or *computing time* [21] that aims at estimating the number of operations between bits required to carry out the algorithm. The binary complexity depends on the form the algorithm and on the size of the integers that it produces. The size of a (Gaussian or real) integer b is measured by $\ell(b)$ where $\ell(\cdot)$ is postulated to have two properties (i) it is additive when two integers are multiplied, $\ell(b_1 b_2) = \ell(b_1)\ell(b_2)$, and (ii) $\ell(b_1 + b_2) = max\{\ell(b_1), \ell(b_2)\}$. It is possible to think of $\ell(b)$ as presenting (approximately) the number of bits (or $Log_2(|b|)$) or the number of decimal digits (or $Log_{10}(|b|)$) (these interpretations obey property (ii) only approximately). We also define $\ell(P(s))$ for an integer polynomial $P(s)$ as the maximal size of (the real and imaginary parts of) its integer coefficients and call it the *integer-size* of the polynomial.

Suppose, we attempt to turn the D-sequence algorithm into an integer algorithm simply by avoiding divisions.

[1] A ring without division operation other than cancellation is called an *integral domain* (also a *unique factorization domain*).

We then assign to $P(s) \in \mathbb{C}[s]$ a sequence of polynomials $F_m(s) = \sum_{k=0}^{m} f_{m,k} s^k$. The initiation would be $A_n(s) = P_o(s)$, $F_n(s) = P_e(s)$ and

$$sF_{n-1}(s) = -a_{n,0}(0)F_n(s) + f_{n,0}A_n(s) \tag{15}$$

followed by recursion for $m = n-1, \ldots, 0$ as follows:

$$\begin{aligned} \gamma_m &= f_{m,0}f_{m+1,1} - f_{m+1,0}f_{m,1}, \\ \delta_m &= f_{m+1,0}f_{m,0} \\ s^2F_{m-1}(s) &= -(\gamma_m s + \delta_m)F_m(s) + f_{m,0}^2 F_{m+1}(s). \end{aligned} \tag{16}$$

Clearly, if $P(s) \in \mathbb{I}_j[s]$ then all $F_m(s) \in \mathbb{I}_j[s]$. So the algorithm is IP. This approach to the Routh array for real polynomials was considered by Barnett [3]. It is possible to go on and deduce for it stability and ZL rules from Theorem 1. The problem is that the outcome will be an inefficient IP stability test because the recursion produces integers of size that grows too rapidly. Using $L_p = \ell(P(s))$ and $\ell_F(m) = \ell(F_m(s))$ to represent integer-sizes for the indicated polynomials, it can be shown that $\ell_F(n-k)$ for $k = 0, 1, \ldots, n$ is given by

$$\ell_F(n-k) = \frac{L_p}{4}\left((2-\sqrt{2})(1-\sqrt{2})^k + (2+\sqrt{2})(1+\sqrt{2})^k\right).$$

This shows that the size of integers increases at an exponential rate with the degree of the polynomial (governed by powers of $1+\sqrt{2}$). So this scheme presents an extremely inefficient integer algorithm even though it has $O(n^2)$ complexity in terms of conventional count of operations. (A good demonstration for why the conventional count of arithmetic operations is not a good measure for the efficiency of an integer algorithm.) The next lemma shows how the size of integers can be restrained.

Lemma 1: Suppose the above division-free algorithm is applied to $P(s) \in \mathbb{I}_j[s]$ and assume it obeys normal conditions (all $F_m(0) \neq 0$). Then

(a) $f_{n,0} | F_{n-2}(s)$ (read "$f_{n,0}$ divides $F_{n-2}(s)$"). Namely, $F_{n-1}(s)/f_{n,0} \in \mathbb{I}_j[s]$.

(b) For $m \leq n-1$, $f_{m,0}^2 | F_{m-2}(s)$. Namely, $F_{m-2}(s)/f_{m,0}^2 \in \mathbb{I}_j[s]$.

Proof: Each $F_m(s)$ is equal to $D_m(s)$ multiplied by a product of positive powers of $d_{n-1,0}, \ldots, d_{m+1,0}$ that accumulate because the divisions that appear in the denominators of (5) are avoided. Since all $d_{k,0} = D_k(0)$ are real and nonzero it follows that all $f_{m,0}$ are real and non zero. In principle, a proof for (a) follows by carrying out the required substitutions. To simplify it, we shall use here (and repeatedly in the following) a congruence technique that is admitted by the GCD property of the recursion. For instance to prove that $F_{n-2}(s)$ has $f_{n,0}$ as a GCD of all its coefficients (the claim in part (a)), it is allowed in the process of substitutions to drop a term, in a sum of terms, as soon as the term is seen to contain the factor $f_{n,0}$. We shall use $\cong$ to denote equality modulo the (claimed) divisor.[2] So to prove part (a) we use $\cong$ to present congruent equality modulo $f_{n,0}$. Begin with $sF_{n-1}(s) \cong -a_{n,0}F_n(s)$ and conclude from here also that $f_{n-1,0} \cong -a_{n,0}f_{n,1}$. Next, $\gamma_{n-1} \cong f_{n-1,0}f_{n,1}$ and $\delta_{n-1} \cong 0$. Therefore $s^2F_{n-2}(s) \cong -f_{n-1,0}f_{n,1}sF_{n-1}(s) + f_{n-1,0}^2F_n(s) \cong f_{n-1,0}F_n(s)\{f_{n,1}a_{n,0} + f_{n-1,0}\} \cong 0$. This completes the proof of part (a).

[2]Note that the claimed exact division follows if the process ends with showing that the dividend $\cong 0$.

To prove part (b), we first rename (to improve clarity) four consecutive polynomials of the F-sequence as: $A(s) = \sum_{k=0}^{K} a_k s_k$, $B(s) = \sum_{k=0}^{K-1} b_k s_k$, $C(s) = \sum_{k=0}^{K-2} c_k s_k$, $D(s) = \sum_{k=0}^{K-3} d_k s_k (K \geq 3)$. It is given that

$$s^2C(s) = -(\gamma_b s + \delta_b)B(s) + b_0^2 A(s) \tag{17}$$

with $\gamma_b = a_1b_0 - a_0b_1$ and $\delta_b = a_0b_0$ and that

$$s^2D(s) = -(\gamma_c s + \delta_c)C(s) + c_0^2 B(s) \tag{18}$$

with $\gamma_c = b_1c_0 - b_0c_1$ and $\delta_c = b_0c_0$. The goal is to prove $b_0^2 | D(s)$. So in the following $\cong$ stands for equality mod b_0^2. Use (17) to write

$$s^2C(s) \cong -(\gamma_b s + \delta_b)B(s). \tag{19}$$

From the above conclude also that

$$c_0 \cong -\gamma_b b_1 - \delta_b b_2 = a_1b_0b_1 - a_0b_1^2 - a_0b_0b_2. \tag{20}$$

Next, by (18) and (19),

$$s^4D(s) \cong (\gamma_c s + \delta_c)(\gamma_b s + \delta_b)B(s) + c_0^2 s^2 B(s). \tag{21}$$

Consider the product $(\gamma_c s + \delta_c)(\gamma_b s + \delta_b)$. Clearly, its free term $\delta_c\delta_b \cong 0$. The coefficient of s consists of $\delta_c\gamma_b \cong -a_0b_0b_1c_0$ plus $\delta_b\gamma_c \cong a_0b_0b_1c_0$. So it too sums up to $\cong 0$. Therefore, (21) simplifies into

$$s^4D(s) \cong s^2B(s)(\gamma_c\gamma_b + c_0^2). \tag{22}$$

Next, evaluate the product $\gamma_c\gamma_b$. Deduce from (19) that

$$c_1 \cong -\gamma_b b_2 - \delta_b b_3 = -a_1b_0b_2 + a_0b_1b_2 - a_0b_0b_3.$$

Therefore,

$$\gamma_c = b_1c_0 - b_0c_1 \cong -a_1b_0b_1^2 + a_0b_1^3 - 2a_0b_0b_1b_2.$$

Multiply it by $\gamma_b = a_1b_0 - a_0b_1$,

$$\gamma_c\gamma_b \cong 2a_0a_1b_0b_1^2 - a_0^2b_1^4 - 2a_o^2b_0b_1^2b_2.$$

Use the evaluation of c_0 in (20) to write

$$c_0^2 \cong a_0^2b_1^4 - 2a_0a_1b_0b_1^3 + 2a_0^2b_0b_1^2b_2.$$

Therefore $\gamma_c\gamma_b + c_0^2 \cong 0$. It follows via (22) that $D(s) \cong 0$. This completes the proof of part (b). ∎

Algorithm 2: The G-sequence. Construct for $P(s) \in \mathbb{C}[s]$ as in (1) a sequence of polynomials $\{G_m(s) = \sum_{k=0}^{m} g_{m,k}s^k, m = n, n-1, \ldots, 0\}$ as follows.

Initiation: Set $A_n(s) = P_o(s)$ and $G_n(s) = P_e(s)$ to the even and odd parts in (3). Set $\epsilon_n = g_{n,0} = p_0'$ and $\gamma_n = a_{n,0} = jp_0''$ and obtain $G_{n-1}(s)$:

$$sG_{n-1}(s) = -\gamma_n G_n(s) + \epsilon_n A_n(s).$$

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

$$\begin{aligned} \epsilon_m &= g_{m,0}^2 \\ \delta_m &= g_{m+1,0}g_{m,0}, \\ \gamma_m &= g_{m,0}g_{m+1,1} - g_{m+1,0}g_{m,1} \end{aligned}$$

$$s^2 G_{m-1}(s) = \frac{-(\gamma_m s + \delta_m) G_m(s) + \epsilon_m G_{m+1}(s)}{\epsilon_{m+1}}. \quad (23)$$

Theorem 2: The G-sequence algorithm is fraction-free over $\mathbb{I}_j$. Namely, if $P(s) \in \mathbb{I}_j[s]$ then all $G_m(s) \in \mathbb{I}_j[s]$ and the whole algorithm can be completed over $\mathbb{I}_j$.

Proof: The algorithm implements recursively the elimination of the exact divisors exposed in Lemma 1. The recursive elimination is admitted due to the GCD nature of the underlying three-term recursion by which if ϵ divides $G_m(s)$, then it divides also all subsequent $G_k(s)$, $k < m$. ∎

The recursive elimination of the factors ϵ_m restrains the growth of integers from exponential rate in the F-sequence to a linear rate as follows.

Proposition 1: Let L_p be the integer-size for $P(s) \in \mathbb{I}_j[s]$ and $\ell_G(m)$ denote the integer-size of the polynomial $G_m(s)$ in the G-sequence. Then,

$$\ell_G(n-k) = 2kL_p, \quad k = 1, \ldots, n. \quad (24)$$

Proof: Clearly, $\ell_G(n) = L_p$, $\ell_G(n-1) = 2L_p$, $\ell_G(n-2) = 4L_p$ (involves division by only $g_{n,0}$). From here on, the recursion becomes uniform and implies that the sizes obey the difference equation $\ell_G(m-1) = 2\ell_G(m) - \ell_G(m+1)$ whose solution for the initial conditions $\ell_G(n-1) = 2L_p, \ell_G(n-2) = 4L_p$ is (24). ∎

The polynomials of the G-sequence differ from polynomials of respective degree in the D-sequence by certain scaling factors as shown next.

Lemma 2: The polynomials of the D-sequence and the G-sequence are related by

$$G_m(s) = \theta_m D_m(s) \quad (25)$$

with $\theta_n = 1$, $\theta_{n-1} = d_{n,0}$ and then, $\theta_m = \theta_{m+1} d_{m+1,0}^2$, i.e.,

$$\theta_m = d_{n,0} d_{n-1,0}^2 \cdots d_{m+1,0}^2\,,\ m = n-2, n-3, \ldots, 0. \quad (26)$$

Proof: Compare the two recursions using induction. $\theta_n = 1$ because by definition $G_n(s) = P_e(s) = D_n(s)$. Therefore, $sG_{n-1}(s) = -a_{n,0}G_n(s) + g_{n,0}A_n(s) = d_{n,0}sD_{n-1}(s)$ shows that $\theta_{n-1} = d_{n,0}$. Use these to prepare $\delta_{n-1} = \theta_n\theta_{n-1}d_{n,0}d_{n-1,0} = d_{n,0}d_{n-1,0}^2\beta_{n-1}$, $\gamma_{n-1} = \theta_n\theta_{n-1}(d_{n,1}d_{n-1,0} - d_{n,0}d_{n-1,1}) = d_{n,0}d_{n-1,0}^2\alpha_{n-1}$, $\epsilon_n = d_{n,0}$ and $\epsilon_{n-1} = \theta_{n-1}^2 d_{n-1,0}^2 = d_{n,0}^2 d_{n-1,0}^2$. Setting them into (23) with $m = n-1$ gives

$$s^2 G_{n-2}(s) = \frac{d_{n,0}d_{n-1,0}^2(\alpha_{n-1}s + \beta_{n-1})d_{n,0}D_{n-1}(s) + d_{n,0}^2 d_{n-1,0}^2 D_n(s)}{d_{n,0}}.$$

Compare the above to (6) with $m = n-1$: $s^2 G_{n-2}(s) = d_{n,0}d_{n-1}^2 s^2 D_{n-2}(s)$, verifies that $\theta_{n-2} = d_{n,0}d_{n-1}^2$. Assume (the induction assumption) that: $\theta_m = \theta_{m+1}d_{m+1,0}^2$ holds for $m = n-2, n-3, \ldots, k$. Use the assumption to prepare: $\delta_k = \theta_{k+1}\theta_k d_{k+1,0}d_{k,0} = \theta_{k+1}\theta_k d_{k,0}^2\beta_k$, $\gamma_k = \theta_{k+1}\theta_k d_{k,0}^2\alpha_k$ and $\epsilon_\kappa = \theta_\kappa^2 d_{\kappa,0}^2$ for $\kappa = k, k+1$. Set these values into (23) and at the end compare the outcome to (6):

$$s^2 G_{k-1}(s) = \frac{-\theta_{k+1}\theta_k d_{k,0}^2(\alpha_m s + \beta_m)\theta_k D_k(s) + \theta_k^2 d_{k,0}^2 \theta_{k+1} D_{k+1}(s)}{\theta_{k+1}^2 d_{k+1,0}^2}$$

$$= \frac{\theta_{k+1}\theta_k^2 d_{k,0}^2}{\theta_{k+1}^2 d_{k+1,0}^2} s^2 D_{k-1}(s) = \theta_k d_{k,0}^2 s^2 D_{k-1}(s).$$

This proves the induction step: $\theta_{k-1} = \theta_k d_{k,0}^2$. ∎

The normal conditions (7) induce normal conditions for the G-sequence given by

$$G_m(0) \neq 0, \quad m = n, n-1, \ldots, 0. \quad (27)$$

Theorem 3: Assume that Algorithm 2 produces for $P(s) \in \mathbb{C}[s]$ as in (1) a normal G-sequence. Then $P(s)$ has ν RHP and $n - \nu$ LHP zeros, where ν is given by

$$\nu = Var\{1, G_{n-1}(0), G_{n-2}(0), \ldots, G_1(0), G_0(0)\}. \quad (28)$$

Proof: By (26), for all $m < n$, $Sgn(\theta_m) = Sgn(d_{n,0})$. Then $g_{m,0} = \theta_m d_{m,0}$ implies $Sgn(g_{m,0}) = Sgn(d_{n,0}d_{m,0})$ and therefore, $Sgn(d_{n,0}g_{m,0}) = Sgn(d_{m,0})$. Set this into (8) to obtain $\nu = Var\{d_{n,0}, d_{n,0}g_{n-1,0}, d_{n,0}g_{n-2,0}, \ldots, d_{n,0}g_{1,0}, d_{n,0}g_{0,0}\}$. The expression (28) follows from here because dividing out the common $d_{n,0}$ from all the entries of the sequence does not affect the count of sign variations. ∎

The implied stability conditions are: $P(s)$ is stable, if and only if,

$$G_m(0) > 0, \quad m = n-1, \ldots, 0. \quad (29)$$

Example 3: Apply Algorithm 2 to $P^{(1)}(s) \in \mathbb{I}_j[s]$ in (10).

$$\begin{aligned}
A_5(s) &= -5i - 2s + \mathrm{j}s^2 + 3s^3 + 2\mathrm{j}s^4 + 3s^5\\
G_5(s) &= -7 - \mathrm{j}s + 3s^2 - 2\mathrm{j}s^3 + 4s^4 - 2\mathrm{j}s^5\\
\epsilon_5 &= -7, \gamma_5 = -5\mathrm{j},\\
G_4(s) &= 19 + 8\mathrm{j}s - 11s^2 + 6\mathrm{j}s^3 - 11s^4\\
\epsilon_4 &= 19^2, \delta_4 = -133, \gamma_4 = 37\mathrm{j},\\
G_3(s) &= 12 - 69\mathrm{j}s - 29s^2 + 45\mathrm{j}s^3\\
\epsilon_3 &= 12^2, \delta_3 = 228, \gamma_3 = 1470\mathrm{j},\\
G_2(s) &= -255 + 87\mathrm{j}s + 171s^2\\
\epsilon_2 &= 255^2, \delta_2 = -3060, \gamma_2 = 16551\mathrm{j},\\
G_1(s) &= 538 + 666\mathrm{j}s\\
\epsilon_1 &= 538^2, \delta_1 = -137190, \gamma_1 = 216636\mathrm{j}\\
G_0(s) &= 2980.
\end{aligned}$$

Notice that all the coefficients and the parameters of the recursion belong to $\mathbb{I}_j$. Use (28) to form $\nu = \{1, 19, 12, -255, 538, 2980\} = 2$. Therefore, by Theorem 3, $P^{(1)}(s)$ has 2 RHP and 3 LHP zeros.

Consider next a $P(s) \in \mathbb{I}_j[s]$ with $p_0 \in \mathbb{I}$. Then $\gamma_n = 0$ and consequently $G_{n-1}(s) = p_0 P_o(s)/s$. It becomes tempting to remove the apparent p_0 factor and use instead $P_o(s)/s$ as the second polynomial of the sequence, if this can be done without compromising the IP property afterwards. We next show that a superfluous p_0 factor can be removed from all subsequent polynomials by just a slight modification at the beginning of the algorithm. For distinction we call the resulting sequence the $\tilde{G}$-sequence and put tildes over all its entries.

Algorithm 3: The $\tilde{\mathrm{G}}$-sequence. Assume $P(s) \in \mathbb{C}[s]$ as in (1) and that $p_0 \in \mathbb{R}$.

Initiation: Set $\tilde{\epsilon}_n = 1$, $\tilde{G}_n(s) = P_e(s)$ and $\tilde{G}_{n-1}(s) = P_o(s)/s$ shown in (11).

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

$$\begin{aligned}\tilde{\epsilon}_m &= \tilde{g}_{m,0}^2,\\ \tilde{\delta}_m &= \tilde{g}_{m+1,0}\tilde{g}_{m,0},\\ \tilde{\gamma}_m &= \tilde{g}_{m,0}\tilde{g}_{m+1,1} - \tilde{g}_{m+1,0}\tilde{g}_{m,1}\\ s^2\tilde{G}_{m-1}(s) &= \frac{-(\tilde{\gamma}_m s + \tilde{\delta}_m)\tilde{G}_m(s) + \tilde{\epsilon}_m\tilde{G}_{m+1}(s)}{\tilde{\epsilon}_{m+1}}. \end{aligned} \quad (30)$$

Theorem 4: The $\tilde{G}$-sequence algorithm is fraction-free over $\mathbb{I}_j$ for $P(s) \in \mathbb{I}_j[s]$ with $p_0 \in \mathbb{I}$. Namely, all $\tilde{G}_m(s) \in \mathbb{I}_j[s]$ and the whole computation can be completed over $\mathbb{I}_j$.

Proof: Assume $P(s) \in \mathbb{I}_j[s]$, $p_0 \in \mathbb{I}$ then $\tilde{G}_n(s), \tilde{G}_{n-1}(s) \in \mathbb{I}_j[s]$ by their definition. Then also $\tilde{G}_{n-2}(s) \in \mathbb{I}_j[s]$ (because its creation involves no division yet). From the next step on, the recursion complies with the terms covered by case (b) in Lemma 1. Therefore subsequent divisions by $\tilde{g}_{m,0}^2$ present cancellation of common factors of the numerator coefficients. ■

Lemma 3: The relations between polynomials of respective degrees in the $\tilde{G}$-sequence and the G-sequence are: $G_n(s) = \tilde{G}_n(s)$ and

$$G_m(s) = p_0\tilde{G}_m(s) \quad m = n-1, \ldots, 0. \quad (31)$$

Proof: Use induction to compare the polynomials $G_m(s)$ and $\tilde{G}_m(s)$ for $m = n, n-1, \ldots$ (in a fashion similar to that in the proof for Lemma 2). ■

Proposition 2: The integer-sizes of the polynomials in the $\tilde{G}$-sequence for $P(s) \in \mathbb{I}_j[s]$ with real p_0 are

$$\ell_{\tilde{G}}(n-k) = (2k-1)L_p, , \quad k = 1, \ldots, n. \quad (32)$$

Proof: $\ell_{\tilde{G}}(m)$ obeys a difference equation similar to that in the proof for Proposition 1 except that now it has to be solved for the initial conditions $\ell_{\tilde{G}}(n-1) = L_p$ and $\ell_{\tilde{G}}(n-2) = 3L_p$. ■

Theorem 5: Assume Algorithm 3 obeys normal conditions ($\tilde{G}_m(0) \neq 0$ for all m) for $P(s) \in \mathbb{C}[s]$ with $p_0 \in \mathbb{R}$. Then $P(s)$ has ν RHP and $n-\nu$ LHP zeros with ν given by

$$\nu = Var\{p_0, \tilde{G}_{n-1}(0), \tilde{G}_{n-2}(0), \ldots, \tilde{G}_1(0), \tilde{G}_0(0)\}. \quad (33)$$

Proof: Use (31) to get $G_m(0) = p_0\tilde{G}_m(0)$ and set it into (28) ■

The implied stability conditions: $P(s)$ is stable, if and only if,

$$Sgn\left(\tilde{G}_m(0)\right) = Sgn\left(p_0\right), \quad m = n-1, \ldots, 0. \quad (34)$$

Note that in difference from the G-sequence where the nice stability conditions (29) hold with no assumption on $p_0 \in C$, in order to have a similar expression for the $\tilde{G}$-sequence, the assumption $p_0 > 0$ has to be added.

IV. FF Tests for Real Polynomials

In this section we consider real polynomials. The goal is to test the stability of

$$P(s) = \sum_{k=0}^{n} p_k s^k \in \mathbb{R}[s]\ ,\ p_n p_0 \neq 0 \quad (35)$$

by an efficient IP algorithm. Clearly both the G-sequence and the $\tilde{G}$-sequence remain IP also for $P(s) \in \mathbb{I}[s]$ where the latter involves somewhat smaller real integers. We first spell out the application of the $\tilde{G}$-sequence algorithm to a real polynomial. Then, in the second part of this section, we shall show that it can be further simplified into a significantly more efficient integer algorithm.

A. Applying the Complex FF Algorithm to a Real Polynomial

When the $\tilde{G}$-sequence algorithm is applied to a $P(s) \in \mathbb{R}[s]$, all the terms with odd powers of s in the polynomials (those that in the complex have purely imaginary coefficients) are absent and all the purely imaginary recursion parameters $\tilde{\gamma}_m$ vanish. For clarity and convenience of subsequent manipulation, we shall refer to the application of $\tilde{G}$-sequence algorithm to a real polynomial as the $\tilde{R}$-sequence. The polynomials of $\tilde{R}$-sequence are even, i.e., have only even powers of s^2. So the actual degree of $\tilde{R}_m(s)$ will be m if m is even and $m-1$ if m is odd, $\tilde{R}_m(s) = \sum_{k=0}^{\ell} \tilde{r}_{m,2k}s^{2k}$ with $m = 2\ell$ or $m = 2\ell+1$.

Algorithm 4: The $\tilde{R}$-sequence. Construct for $P(s)$ in (35) a sequence $\{\tilde{R}_m(s) = \sum_{i=0}^{\lfloor m/2 \rfloor} \tilde{r}_{m,2i}s^{2i}, m = n, \ldots, 0\}$ as follows.

Initiation: Set $\epsilon_n = 1$ and

$$\begin{aligned}\tilde{R}_n(s) &= \tfrac{P(s)+P(-s)}{2} = p_0 + p_2 s^2 + \cdots\\ \tilde{R}_{n-1}(s) &= \tfrac{P(s)-P(-s)}{2s} = p_1 + p_3 s^2 + \cdots\end{aligned}. \quad (36)$$

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

$$\begin{aligned}\epsilon_m &= \tilde{r}_{m,0}^2,\\ \delta_m &= \tilde{r}_{m+1,0}\tilde{r}_{m,0}\\ s^2\tilde{R}_{m-1} &= \frac{-\delta_m\tilde{R}_m(s) + \epsilon_m\tilde{R}_{m+1}(s)}{\epsilon_{m+1}}.\end{aligned} \quad (37)$$

The next theorem and proposition follow from Theorem 4 and Proposition 2, respectively, simply because the $\tilde{R}$-sequence is the application of the $\tilde{G}$-sequence to a real polynomial.

Theorem 6: The $\tilde{R}$-sequence algorithm is fraction-free over $\mathbb{I}$. Namely, if $P(s) \in \mathbb{I}[s]$ then all $\tilde{R}_m(s) \in \mathbb{I}[s]$ and the whole algorithm can be completed over $\mathbb{I}$.

Proposition 3: The integer-size of the polynomials in the $\tilde{R}$-sequence for $P(s) \in \mathbb{I}[s]$ is

$$\ell_{\tilde{R}}(n-k) = (2k-1)L_p, \quad k = 1, \ldots, n. \quad (38)$$

Theorem 7: Assume that Algorithm 4 obeys for $P(s)$ in (35) normal conditions (all $\tilde{R}_m(0) \neq 0$). Then $P(s)$ has ν RHP and $n-\nu$ LHP zeros, where ν given by

$$\nu = Var\{p_0, \tilde{R}_{n-1}(0), \ldots, \tilde{R}_1(0), p_n\}. \quad (39)$$

Proof: By Theorem 5, we can write for the $\tilde{R}$-sequence

$$\nu = Var\{p_0, \tilde{R}_{n-1}(0), \ldots, \tilde{R}_1(0), \tilde{R}_0(0)\}. \quad (40)$$

Notice that $\tilde{R}_1(s) = \tilde{r}_{1,0}$ and $\tilde{R}_0(s) = \tilde{r}_{0,0}$. The replacement of $\tilde{r}_{0,0}$ with p_n is admitted because $Sgn\left(\tilde{r}_{0,0}\right) = Sgn\left(p_n\right)$. This is a tricky observation that we shall prove later using a forthcoming relation (48), see Remark 4. ■

The stability conditions correspond to a same sign in (39), i.e., $\nu = 0$. If $p_0 > 0$ may be added as a presumption, then the set of necessary and sufficient conditions for stability becomes

$$p_n > 0, p_0 > 0\ ,\ \tilde{R}_m(0) > 0,\ m = n-1, \ldots, 1. \qquad (41)$$

Example 4: To illustrate Algorithm 4 we apply it to $P^{(2)}(s) \in \mathbb{I}[s]$ in (13). The resulting $\tilde{R}$-sequence (where the values of $\delta_m,\ \epsilon_m,\ m = 5, \ldots, 1$ are omitted for brevity) is:

$$\begin{array}{l} \tilde{R}_6(s) = 30 + 84s^2 + 31s^4 + 2s^6 \\ \tilde{R}_5(s) = 71 + 66s^2 + 10s^4 \\ \tilde{R}_4(s) = 282864 + 134971s^2 + 10082s^4 \\ \tilde{R}_3(s) = 509844432 + 118555872s^2 \\ \tilde{R}_2(s) = 224801338963 + 32754177458s^2 \\ \tilde{R}_1(s) = 8606631150810 \\ \tilde{R}_0(s) = 48010396480200 \end{array} .$$

It is apparent that the conditions (41) hold. Therefore $P^{(2)}(s)$ is stable.

B. Optimal FF Test for a Real Polynomial

In this subsection we derive a far better FF stability test for real integer polynomials based on the following algorithms.

Algorithm 5: The R-sequence. Construct for $P(s) \in \mathbb{R}[s]$ as in (35) a sequence of real polynomials $\{R_m(s) = \sum_{i=0}^{\lfloor m/2 \rfloor} r_{m,2i} s^{2i}, m = n, \ldots, 0\}$ as follows.

Initiation: Set $\eta_n = 1$, $\eta_{n-1} = 1$ and

$$\begin{array}{rcl} R_n(s) & = & \frac{P(s)+P(-s)}{2} = p_0 + p_2 s^2 + \cdots \\ R_{n-1}(s) & = & \frac{P(s)-P(-s)}{2s} = p_1 + p_3 s^2 + \cdots \end{array} . \qquad (42)$$

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

$$\begin{array}{l} \eta_{m-1} = r_{m,0} \\ s^2 R_{m-1}(s) = \dfrac{-r_{m+1,0} R_m(s) + r_{m,0} R_{m+1}(s)}{\eta_{m+1}}. \end{array} \qquad (43)$$

Lemma 4: Consider four successive polynomials $F_m(s) = \sum_{k=0}^{K-m} f_{m,2k} s^{2k} \in \mathbb{I}[s],\ m = 0, 1, 2, 3$ (for any $K \geq 3$) that obey a division-free version of (43). Namely, assume they are related by

$$s^2 F_{m+2}(s) = -f_{m,0} F_{m+1}(s) + f_{m+1,0} F_m(s). \qquad (44)$$

Then $f_{1,0} | F_4(s)$. Namely, $F_4(s)/f_{1,0} \in \mathbb{I}[s]$.

Proof: We carry out the involved substitutions using $\cong$ to present congruent equality modulo $f_{1,0}$. Obtain from the first recursion step

$$s^2 F_2(s) = -f_{0,0} F_1(s) + f_{1,0} F_0(s) \cong -f_{0,0} F_1(s).$$

Get from here also $f_{2,0} \cong -f_{0,0} f_{1,2}$. Obtain from the second recursion step

$$s^2 F_3(s) = -f_{1,0} F_2(s) + f_{2,0} F_1(s) \cong f_{2,0} F_1(s)$$

that also implies $f_{3,0} \cong f_{1,2} f_{2,0}$. Take the third recursion step and multiply its two sides by s^2. Then set into it the above congruent values for $s^2 F_3(s)$ and $f_{3,0}$

$$\begin{array}{rl} s^4 F_4(s) = & -f_{2,0} s^2 F_3(s) + f_{3,0} s^2 F_2(s) \\ \cong & -f_{2,0}(f_{2,0} F_1(s)) + (f_{1,2} f_{2,0})(-f_{0,0} F_1(s)) \\ = & f_{2,0} F_1(s)(-f_{2,0} - f_{1,2} f_{0,0}) \cong 0 \end{array}$$

where the last $\cong 0$ follows from the congruent value of $f_{2,0}$. We reached $F_4(s) \cong 0$ that means that $f_{1,0}$ is a common divisor of all the coefficients of $F_4(s)$. ■

Similar to the reasoning that led from Lemma 1 to Theorem 2, Lemma 4 implies

Theorem 8: The R-sequence algorithm is fraction-free over $\mathbb{I}$. Namely, if $P(s) \in \mathbb{I}[s]$ then all the polynomials $R_m(s) \in \mathbb{I}[s]$ and the whole algorithm can be completed over $\mathbb{I}$.

Proposition 4: Let $\ell_R(m)$ denote the integer-size of $R_m(s)$ in the R-sequence for $P(s) \in \mathbb{I}[s]$. Then,

$$\ell_R(n-k) = kL_p, \quad k = 1, \ldots, n. \qquad (45)$$

Proof: Inspection at the algorithm reveals that $\ell_R(n) = \ell_R(n-1) = L_p$, then $\ell_R(n-2) = 2L_p, \ell_R(n-3) = 3L_p$. From here on, the recursion takes its steady form and the difference equation becomes $\ell_R(n-k) = \ell_R(n-k+1) + \ell_R(n-k+2) - \ell_R(n-k+3)$. This is a third order difference equation whose solution for the given initial conditions is (45). ■

Lemma 5: The polynomials of the R-sequences and the $\tilde{R}$-sequences are related by $\tilde{R}_n(s) = R_n(s)$, $\tilde{R}_{n-1}(s) = R_{n-1}(s)$ and

$$\tilde{R}_m(s) = r_{m+1,0} R_m(s)\ , \quad m = n-2, \ldots, 0. \qquad (46)$$

Proof: Compare $\tilde{R}_m(s)$ and $R_m(s)$ for $m = n, n-1, \ldots$ using induction (like in the proof for Lemma 2). ■

It follows that the R-sequence inherits the normal conditions:

$$R_m(0) \neq 0, \quad m = n, \ldots, 0. \qquad (47)$$

The next lemma states a simple but very useful relation.

Lemma 6: The last term of a normal R-sequence obeys the relation

$$r_{0,0} = r_{1,0} p_n. \qquad (48)$$

Proof: Use (43) to obtain the next relation between the leading coefficients $r_{m,m}$ of the polynomials

$$r_{m-1,m-1} = \frac{r_{m,0} r_{m+1,m+1}}{\eta_{m+1}}.$$

Notice that $r_{m,m} = 0$ for odd m. Start with $r_{0,0}$ and proceed with $r_{m,m}$'s of even m upward: $r_{0,0} = r_{1,0} r_{2,2} / r_{3,0}$, substitute into it $r_{2,2} = r_{3,0} r_{4,4} / r_{5,0}$ gives $r_{0,0} = r_{1,0} r_{4,4} / r_{5,0}$, substitute in the latter $r_{4,4} = r_{5,0} r_{6,6} / r_{7,0}$ gives $r_{0,0} = r_{1,0} r_{6,6} / r_{7,0}$ and so forth till reaching at the last step, using $\eta_n = \eta_{n-1} = 1$,

$$r_{0,0} = \begin{cases} r_{1,0} r_{n,2m} & n = 2m \\ r_{1,0} r_{n-1,2m} & n = 2m+1. \end{cases}$$

But $p_n = r_{n,2m}$ when $n = 2m$ and $p_n = r_{n-1,2m}$ when $n = 2m+1$ (take a look at (42)). Thus (48) holds for both parities of n. ■

Theorem 9: Assume the R-sequence obeys for $P(s) \in \mathbb{R}[s]$ as in (35) the normal conditions (47). Then $P(s)$ has ν RHP and $n-\nu$ LHP zeros, where ν is given by

$$\nu = Var\{p_0, R_{n-1}(0), R_{n-1}(0)R_{n-2}(0), \ldots, R_2(0)R_1(0), p_n\}. \tag{49}$$

Proof: Use (40) and Lemma 5 to write

$$\nu = Var\{p_0, R_{n-1}(0), R_{n-1}(0)R_{n-2}(0), \ldots, R_1(0)R_0(0)\} \tag{50}$$

where $R_1(0) = r_{1,0}$ and $R_0(0) = r_{0,0}$. But (48) implies $r_{1,0}r_{0,0} = r_{1,0}^2 p_n$. Namely, $Sgn\,(r_{1,0}r_{0,0}) = Sgn\,(p_n)$. Therefore, the last entry in (50) can be replaced by p_n. ■

Remark 3: The ability to determine ZL with (49) instead of (50) is valuable because it uses p_n that is known from the outset instead of $r_{0,0}$ formed at the last step of the recursion. From the perspective of integers, saving the last step also reduces the largest integer required for completing the test.

The stability conditions for the R-sequence correspond to $\nu = 0$ in (50) or (49). The product of adjacent $R_m(0)$ pairs in these expressions can be decoupled. If $p_0 < 0$ then the remaining necessary and sufficient conditions for stability are $(-1)^k R_{n-k}(0) > 0,\ k = 1, \ldots, n$ while if $p_0 > 0$ (a worthy convenient presumption) then necessary and sufficient conditions for stability are given by the more pleasant set:

$$p_0 > 0, p_n > 0, R_m(0) > 0\ ,\ m = n-1, \ldots, 1. \tag{51}$$

Remark 4: In the proof of Theorem 7 for the $\tilde{R}$-sequence, the property $Sgn\,(\tilde{r}_{0,0}) = Sgn\,(p_n)$ was used with a promise for a forthcoming proof. Use Lemma 5 to write $\tilde{R}_0(s) = \tilde{r}_{0,0} = r_{1,0}r_{0,0}$. Then use (48) to write it as $\tilde{r}_{0,0} = r_{1,0}^2 p_n$ implies the required property.

Example 5: To illustrate the R-sequence, we apply Algorithm 5 to $P^{(2)}(s) \in \mathbb{I}[s]$ in (13) and obtain

$$\begin{aligned} R_6(s) &= 30 + 84s^2 + 31s^4 + 2s^6 \\ R_5(s) &= 71 + 66s^2 + 10s^4 \\ R_4(s) &= 3984 + 1901s^2 + 142s^4 \\ R_3(s) &= 127973 + 29758s^2 \\ R_2(s) &= 1756631 + 255946s^2 \\ R_1(s) &= 4899510 \\ R_0(s) &= 9799020 \end{aligned} \quad .$$

By Theorem 9, or by the conditions (51), the polynomial is stable. Notice the reduction in size of integers in comparison with Example 4. It illustrates reasonably well the near factor of two gain in the size of integers of the R-sequence compared to the $\tilde{R}$-sequence featured in (45) and (38), respectively.

Jeltsch mentions [19] that there exists another Routh array for real polynomials that can be deduced to be too FF via the subresultant polynomial remainder sequence theory [21]. It is mentioned there quite casually and dismissed as less attractive because it involves integers of double size compared to the optimal Routh array. It can be readily realized that this alternative array corresponds to our $\tilde{R}$-sequence (up to our reversed setting). Here the FF property of the $\tilde{R}$-sequence is established directly, it is complemented with stability and zero location rules and its relation to the R-sequence is presented. Not less important is that the paper now shades light on the somewhat enigmatic dual FF array mentioned briefly in [19] and puts it into an interesting perspective. Namely, it can be regarded as the 'projection' of the complex FF Routh test on the real case. The hierarchy begins with the G-sequence algorithm that is the best FF Routh test for a general Gaussian integer polynomial. The G-sequence, or more precisely its simplification into the $\tilde{G}$-sequence, inherits for real polynomials the $\tilde{R}$-sequence that is FF but no longer optimal for real integer polynomials. For the real case, further systematic extraction of common integers is possible and yields the nearly half-sized integer R-sequence. So far the relative performances of these integer Routh tests were compared just by the size of the integers they produce. We next attend to measure their efficiency by binary complexity.

V. Complexity

The efficiency of an integer algorithm is measured most adequately by what is called "binary complexity" [20] or "computing time" [21]. The measure estimates the number of binary operations required to carry it out. The required calculations for the FF Routh algorithms begin by regarding the $\ell(m)$ measures found for the G-sequence, the $\tilde{R}$-sequence and the R-sequence in (24), (38) and (45), respectively, as measuring the size of the integer coefficients in bits. Then, a count of the overall number of binary flops (one flop means one real multiplication plus one addition) for each algorithm can be carried out, taking into account the structure of the respective recursion, the number of coefficients at each step (that descend with the degrees from step to step) and using expressions for the cost of multiplication or cancellation (exact division) of two integers (e.g., the multiplication of two integers of lengths $\ell_1,\ \ell_2$, requires about $\ell_1\ell_2$ binary flops, see similar counts carried out for the integer unit-circle test in [16]). It is relatively easy to realize that the counts for each of these algorithms must be of order $O(\alpha L_p^2 n^4)$, a notation that is used to mean $\alpha L_p^2 n^4 +$ terms with lower power of L_p and/or n. Skipping tedious details, the α for the R-sequence is $\alpha_R = 1/12$. The binary complexity of the $\tilde{R}$-sequence is more than 10 times higher, with $\alpha_{\tilde{R}} = 11/12$. For the G-sequence $\alpha_{\tilde{G}} = 30/12$ (recall that its coefficients are either real or purely imaginary, never both). The higher binary complexity of the G-sequence in comparison to the $\tilde{R}$-sequence is contributed by the extra multiplier in the numerator of the recursion (23) and by the doubling of the number of nonzero coefficients (in the real case only powers of s^2 participate).

The above efficiency counts disregard possible common factors in the coefficients of $P(s)$. Common integer factor grow at an exponential rate through the recursion. So removing apparent integer common factors may results an impressive reduction in integers size. If suitable, the systematic way to ensure the removal of all common factors is the preliminary running of an extraneous integer GCD algorithm over all the coefficients of the tested polynomial. Latent common factors are not expected when one or more of the coefficients involve literal parameters.

The G-sequence is the most efficient integer algorithm for a general $P(s) \in \mathbb{I}_j[s]$. This is so because (16) is the sim-

plest Euclidean degree-reducing step between two complex even polynomials that remains over Gaussian integers, and Lemma 1 presents all the possible cancelations of common factors that its repeated application produces. Consequently, the recursion involves the smallest possible size of integers. The claim of least size of integers assumes a fully complex coefficient polynomial, namely, that all the coefficients p'_k and p''_k, $k = 1, \ldots, n$ participate. For instance, when the supposed to be complex p_0 is real, we saw that the simpler $\tilde{G}$-sequence becomes possible. It was also assumed that there is access to the real and imaginary parts p'_k and p''_k of each coefficient p_k. This may not be the case in a situation where access to some of the coefficients is granted only through the symbol $p_k \in \mathbb{C}$ and its conjugate $\bar{p}_k$. The remedy for such a situation is to use $p_k + \bar{p}_k = 2p'_k$ and $p_k - \bar{p}_k = 2jp''_k$, i.e., to initiate the algorithm with $2P_e(s)$ and $2P_o(s)$ instead. The consequence is that subsequent $G_{n-k}(s)$'s are scaled up by factors of 2^k, $k = 1, 2, \ldots, n$. Similarly (and without corresponding reservations), the R-sequence is optimal for a real integer polynomial because (44) is the simplest Euclidean degree reducing step between two real even polynomials that remains over integers, and Lemma 4 exposes all the common factors that its repeated use creates.

VI. Concluding Remarks

The paper presented two different fraction-free Routh tests. One is the G-sequence algorithm that is most efficient for Gaussian integer polynomials. The second is the R-sequence that is about 10 times more efficient for an integer polynomial (in binary complexity) than using the G-sequence for a real polynomial. These FF tests have interesting relations with stability determinants of Bezout Hermite Sylvester and Hurwitz matrices that due to space restrictions were not covered in the current scope. At the same time, the fact that the paper has established the FF Routh tests without reliance on matricial stability theory contributed to making the presentation self contained and demonstrates once again that efficient stability tests and their possible relation to matricial stability criteria are two separable issues.

Following the terminology suggested in [17], the G-sequence in this paper can be regarded as the *continuous equivalent* of the FF Bistritz test for Gaussian integer polynomials appearing in [15] (that also underlies earlier 2-D tabular stability tests for discrete systems cited there). Similarly, the R-sequence is the *continuous equivalent* of the FF IP Bistritz test for real integer polynomials in [16]. Thus, the work presented in this paper provides further justification to the perception that led Jury and Mansour [17] to name the Bistritz test as the discrete equivalent of the Routh test and distinguish it from the previous Jury-Marden stability test that provides only a *discrete counterpart* for the Routh test. The emerging new distinction is that the modified Jury test in [22] (presented by Jury already earlier in several variants summarized in the type C tests in [10]), which is actually *the* FF test for the SCMJ class of stability tests for discrete systems, does not have two forms, one for Gaussian integers and a different and simpler one for real integers.

The FF Routh algorithms are in general more suitable for today's computer algebra systems and in conjunction with a symbolic language software provide useful design tools for continuous-time systems. For example, they may be used to determine stability range for designable parameters similar to the use of the FF Bistritz test in [16]. In addition, since integer arithmetics is exact, these FF tests introduce definiteness to the Routh test because in its original form rounding error may affect the sign rules when testing high degree or ill-conditioned polynomials. Note that accuracy can be granted also for polynomial with decimal coefficients by first scaling them by a proper power of 10 into an integer polynomial. More applications are expected to arise based on the fact that these algorithms remain fraction-free also for polynomials with coefficients over more general Gaussian-integer-like and real-integer-like integral domains.

Appendix
Proof for Theorem 1

Assume Algorithm 1 obeys for $P(s) \in \mathbb{C}[s]$ (1) normal conditions. Then the sequence $\{D_m(s)\}_{m=0}^{n}$ of even polynomials is well defined and obeys the recursion (6). We shall use the next properties.

— *Property 1*. $D_m(\mathrm{j}\omega) = \rho_m(\omega)$ is a real polynomial in ω.

— *Property 2*. Two successive polynomials of the sequence can not vanish at a common finite zero on $\mathcal{I}$.

— *Property 3*. No two successive polynomials in the D-sequence can be degree deficient.

Proof: Properties 1 was explained in Remark 2. Property 2 is implied by normality and the GCD property of the recursion discussed in Remark 1. It remains to prove property 3. The assumption that $p_n \neq 0$ implies that $D_n(s)$ and $D_{n-1}(s)$ can not be both degree deficient. This can be realized as follows.

$$d_{n,n} = \begin{cases} p'_n & \text{even } n \\ \mathrm{j}p''_n & \text{odd } n \end{cases}, \quad a_{n,n} = \begin{cases} \mathrm{j}p''_n & \text{even } n \\ p'_n & \text{odd } n. \end{cases}$$

Therefore,

$$d_{n-1,n-1} = \begin{cases} \mathrm{j}\left(-\frac{p''_0}{p'_0}p'_n + p''_n\right) & \text{even } n \\ -\frac{p''_0}{p'_0}p''_n + p'_n & \text{odd } n. \end{cases}$$

For n even, if $p''_n = 0$ then $d_{n,n} = p'_n \neq 0$ else if $p'_n = 0$ then $d_{n-1,n-1} = \mathrm{j}p''_n \neq 0$. For n odd if $p'_n = 0$ then $d_{n,n} = \mathrm{j}p''_n \neq 0$ else if $p''_n = 0$ then $d_{n-1,n-1} = p'_n \neq 0$. Subsequently, two consecutive polynomials $D_m(s)$ and $D_{m-1}(s)$, $m < n$ can not be both degree deficient because this degree deficiency (that can be regarded as a common zero at ∞) would propagate downward and imply premature termination of the sequence, in contradiction to the assumption that normal conditions hold. ■

Lemma A: Assume that Algorithm 1 produces for $P(s)$ a normal D-sequence. Associate to each pair of adjacent polynomials, the complex polynomial $F_m(s) = D_m(s) + sD_{m-1}(s)$ for $m = n, \ldots, 1$. Let $\zeta_R[F(s)]$ denote the number of zeros of the polynomial $F(s)$ in the open RHP, $\mathcal{R}e\ s > 0$. Then for $m = n-1, \ldots, 0$

$$\zeta_R[F_{m+1}(s)] = n_-\{\beta_m\} + \zeta_R[F_m(s)]. \qquad \text{(A1)}$$

Proof of Lemma A: Write $F_{m+1}(s)$, for any $0 \leq m \leq n-1$, as follows:

$$\begin{aligned} F_{m+1}(s) &= D_{m+1}(s) + sD_m(s) \\ &= [(\alpha_m + 1)s + \beta_m]D_m(s) + s^2 D_{m-1}(s). \end{aligned}$$

By adding and subtracting $sD_{m-1}[(\alpha_m + 1)s + \beta_m]$ obtain

$$F_{m+1}(s) = [(\alpha_m + 1)s + \beta_m]F_m(s) - sD_{m-1}[\alpha_m s + \beta_m].$$

The above can be written as follows:

$$F_{m+1}(s) = [(\alpha_m + 1)s + \beta_m]F_m(s)[1 - Q^{\{m\}}(s)] \quad \text{(A2)}$$

with

$$Q^{\{m\}}(s) = \frac{(\alpha_m s + \beta_m)sD_{m-1}(s)}{[(\alpha_m + 1)s + \beta_m]F_m(s)}. \quad \text{(A3)}$$

We aim at showing that $1 - Q^{\{m\}}(s)$ does not vanish on the boundary of the RHP that can be thought of as the limit $\Gamma = \lim_{\rho \to \infty} \Gamma_\rho$ of $\Gamma_\rho = \mathcal{I}_\rho \cup \mathcal{A}_\rho$ in which $\mathcal{I}_\rho = \{s | s = j\omega, \omega \in [-\rho, \rho]\}$ and $\mathcal{A}_\rho$ denotes an arc of radius ρ in the RHP supported by $\mathcal{I}_\rho$.

Factor (A3) into $Q^{\{m\}}(s) = Q_1^{\{m\}}(s)Q_2^{\{m\}}(s)$ where

$$\begin{aligned} Q_1^{\{m\}}(s) &= \frac{\alpha_m s + \beta_m}{(\alpha_m + 1)s + \beta_m} \\ Q_2^{\{m\}}(s) &= \frac{sD_{m-1}(s)}{F_m(s)} = \frac{1}{1 + \frac{D_m(s)}{sD_{m-1}(s)}}. \end{aligned}$$

Examine first $Q_1^{\{m\}}(s)$. Since β_m is real and nonzero and α_m is imaginary, it follows that $Q_1^{\{m\}}(0) = 1$, that for all $0 \neq s \in \mathcal{I}_\rho$, $|Q_1^{\{m\}}(s)| < 1$, and that $\lim_{s \to \infty} |Q_1^{\{m\}}(s)| = |\alpha_m/(\alpha_m + 1)| < 1$. Therefore $|Q_1^{\{m\}}(s)| < 1$ for all $0 \neq s \in \Gamma$.

Next, consider $Q_2^{\{m\}}(s)$. Clearly $Q_2^{\{m\}}(0) = 0$ and $\lim_{s \to \infty} Q_2^{\{m\}}(s) = d_{m-1,m-1}/(d_{m,m} + d_{m-1,m-1})$. By property 3, $d_{m-1,m-1}$ and $d_{m,m}$ may vanish but not simultaneously, also $d_{m,m} + d_{m-1,m-1} \neq 0$ (sum of a real number and a pure imaginary number). So $\lim_{s \to \infty} |Q_2^{\{m\}}(s)| \leq 1$. Else, for all $0 \neq s \in \mathcal{I}_\rho$, $Q_2^{\{m\}}(s)$ has the form $1/(1 - \mathrm{j}Y)$ where $Y = \rho_m(\omega)/(\omega\rho_{m-1}(\omega))$ is real and finite. Therefore, $|Q_2^{\{m\}}(s)| \leq 1$ for all $s \in \Gamma_\rho$. Combining the analysis of the two factors of $Q^{\{m\}}(s)$, we showed that $|Q^{\{m\}}(s)| < 1$ for all $s \in \Gamma$. (Notice that we showed the strict inequality also at $s = 0, \infty$). It follows that the factor $1 - Q^{\{m\}}(s)$ in (A2) does not vanish for all $s \in \Gamma$. So the Argument Principle (or Rouché's theorem) can be applied to conclude that

$$\zeta_R [F_{m+1}(s)] = \zeta_R [(\alpha_m + 1)s + \beta_m] + \zeta_R [F_m(s)].$$

The zero of the first term is in the RHP if $\beta_m < 0$ and in the LHP if $\beta_m > 0$ (recall that $\alpha_m \in \mathcal{I}$ and $0 \neq \beta_m \in \mathbb{R}$). Therefore $\zeta_R [(\alpha_m + 1)s + \beta_m] = n_-\{\beta_m\}$. ∎

Proof of Theorem 1: Apply (A1) recursively (from bottom upward) to obtain that $\zeta_R [F_1(s)] = n_-\{\beta_0\} + \zeta_R [F_0(s)] = n_-\{\beta_0\}$, $\zeta_R [F_2(s)] = n_-\{\beta_1\} + \zeta_R [F_1(s)] = n_-\{\beta_1, \beta_0\}$, and so forth, $\zeta_R [F_m(s)] = n_-\{\beta_{m-1}, \ldots, \beta_0\}$, till

$$\zeta_R [F_n(s)] = n_-\{\beta_{n-1}, \ldots, \beta_0\}.$$

It remains to show that $\zeta_R [P(s)] = \zeta_R [F_n(s)]$. Write the relation between $P(s) = P_o(s) + P_e(s)$ and $F_n(s) = D_n(s) + sD_{n-1}(s)$ as follows:

$$P(s) = (\alpha_n + 1)F_n(s)(1 - Q(s))$$

where

$$Q = \frac{\alpha_n sD_{n-1}(s)}{(\alpha_n + 1)(D_n(s) + sD_{n-1}(s))} = \left(\frac{\alpha_n}{1 + \alpha_n}\right) Q_2^{\{n\}}(s).$$

The first factor is independent of s and has absolute value less than 1. The second factor $Q_2^{\{n\}}(s)$ is similar to $Q_2^{\{m\}}(s)$ $m < n$ analyzed above. Thus $|Q_2^{\{n\}}(s)| \leq 1$ on Γ. Hence $|Q(s)| < 1$ on Γ. Invoking again the Argument Principle, it follows that $\zeta_R [P_n(s)] = \zeta_R [F_n(s)]$. This completes the proof of (9) that is equivalent to (8). ∎

REFERENCES

[1] E. Routh, *A Treatise on the Stability of a Fiven State of Motion*. London, U.K.: MacMillan & Co., 1877.

[2] A. Hurwitz, "Über die Bedingungen unter welchen eine Gleichung nur Wurzeln mit negativen reellen Teilen besitzt," *Math. Ann.*, vol. 46, pp. 273–284, 1895, (English translation by H. G. Bergmann in *Selected Papers in Mathematical Trends in Control Theory*, R. Bellman and R. Kalaba, Eds. New York: Dover, 1964, pp. 70–82.).

[3] S. Barnett, *Polynomials and Linear Control Systems*. Basel, NY, USA: Marcel Decker, 1983.

[4] M. Marden, *The Geometry of the Zeros of a Polynomial in a Complex Variable*. New York, NY, USA: American Math. Society, 1949.

[5] F. R. Gantmacher, *Matrix Theory*. New York, NY, USA: Chelsea, 1959, vol. II, ch. XV.

[6] H. S. Wall, *Analytic Theory of Continued Fractions*. New York, NY, USA: Chelsea, 1948, ch. 10.

[7] E. Frank, "On the zeros of polynomials with complex coefficients," *Bull. Amer. Math. Soc.*, vol. 52, pp. 144–157, 1946.

[8] V. Belevitch, *Classical Network Theory*. San Francisco, CA, USA: Holden-Day, 1968.

[9] D. Pal and T. Kailath, "Displacement structure approach to singular root distribution problems: The imaginary axis case," *IEEE Trans. Circuits Syst. I*, vol. 41, no. 2, pp. 138–148, 1992.

[10] Y. Bistritz, "Reflections on Schur-Cohn matrices and Jury-Marden tables and classification of related unit-circle zero location criteria," *Circuits Syst. Signal Process.*, vol. 51, no. 1, pp. 111–136, 1996.

[11] H. C. Reddy and P. K. Rajan, "All pass function based stability test for continuous time systems," in *Proc. IEEE Int. Symp. Circuits and Systems*, 1985, pp. 824–826.

[12] A. Lepschy, G. A. Mian, and U. Viaro, "Splitting of some s-domain stability-test algorithms," *Int. J. Control*, vol. 50, pp. 2237–2247, 1989.

[13] W. Krajewski, A. Lepschy, G. A. Mian, and U. Viaro, "A unifying frame for stability-test algorithms for continuous-time systems," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 290–296, 1990.

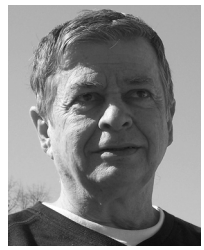
[14] H. Lev-Ari, Y. Bistritz, and T. Kailath, "Generalized Bézoutians and families of efficient zero-location procedures," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 170–186, Feb. 1991.

[15] Y. Bistritz, "On testing stability of 2-D discrete systems by a finite collection of 1-D stability sets," *IEEE Trans. Circuits Syst. I*, vol. 49, no. 11, pp. 1634–1638, 2002.

[16] Y. Bistritz, "An efficient integer-preserving stability test for discrete-time systems," *Circuits Syst. Signal Process.*, vol. 23, no. 3, pp. 195–213, 2004.

[17] E. I. Jury and M. Mansour, "On the terminology relationship between continuous and discrete systems criteria," *Proc. IEEE*, vol. 73, no. 4, p. 884, 1985.

[18] Y. Bistritz, "Zero location of polynomials with respect to the unit-circle unhampered by nonessential singularities," *IEEE Trans. Circuits Syst. I*, vol. 49, no. 3, pp. 305–314, 2002.
[19] R. Jeltsch, "An optimal fraction free Routh array," *Int . J. Control*, vol. 30, no. 4, pp. 653–660, 1979.
[20] S. Basu, R. Pollack, and M. F. Roy*, Algorithms in Real Algebraic Geometry*, 2nd ed. New York, NY, USA: Springer-Verlag, 2008.
[21] W. S. Brown, "The subresultant PRS algorithm," *ACM Trans. Math. Software*, vol. 4, pp. 237–249, 1978.
[22] K. Premaratne and E. I. Jury, "On the Bistritz tabular form and its relationship with the Schur–Cohn minors and inner determinants," *J. Franklin Inst.*, vol. 330, no. 1, pp. 165–182, 1993.



Yuval Bistritz received the B.Sc. degree in physics and the M.Sc. and Ph.D. degrees in electrical engineering from Tel Aviv University, Tel Aviv, Israel, in 1973, 1978, and 1983, respectively.

From 1979 to 1984 he held various assistant and teaching positions in the department of Electrical Engineering, Tel Aviv University, and in 1987 he joined the department as a Faculty Member. From 1984 to 1986 he was a research scholar in the Information System Laboratory, Stanford University, Stanford, CA, USA, working on fast signal processing algorithms. From 1986 to 1987 he was with AT&T Bell Laboratories, Murray Hill, NJ, USA, and from 1994 to 1996 with DSP Group, Santa Clara, CA, USA, working on speech processing algorithms. His research interests are in the area of digital signal processing and system theory.