12-1-1986

A Routh-like algorithm in system identification and adaptive control

Chaouki T. Abdallah

F. L. Lewis

B. L. Stevens

Proceedings of 25th Conference
on Decision and Control
Athens, Greece • December 1986

A ROUTH-LIKE ALGORITHM IN SYSTEM IDENTIFICATION AND ADAPTIVE CONTROL

C. Abdallah and F. L. Lewis*

School of Electrical Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0250
(404) 894-2994

B. L. Stevens

Advanced Research Organization
Lockheed-Georgia Company
Marietta, Georgia 30063
(404) 424-2204

Abstract

This paper introduces algorithms to be used for finding the Cauchy index of a transfer function and for solving the Diophantine equation. The algorithms are similar to the Routh-Hurwitz array in structure and are simple to implement. The first algorithm also provides a simple check for the presence of almost common factors between two polynomials, and so is useful as a computationally inexpensive alternative to the Sylvester determinant. It also finds the almost common factor, and so is potentially useful as an alternative to [8] in implementing self-tuning regulators for nonminimum-phase systems [6].

I. Introduction

In linear system theory, the Cauchy index of a given transfer function plays an important role. In particular, in model-order reduction and identification problems, the structural changes can be monitored by finding the Cauchy index at each stage [1,4]. In [1], it is suggested that pole-zero cancellations in the estimated transfer function could be avoided (if the plant is prime) if the estimates are initialized to have the same Cauchy index as the plant. The methods presented so far to calculate the Cauchy index rely on its definition or on a variation of it that necessitates the evaluation of the signatures of some matrices [3,4].

It is known that problems arise in a self-tuning regulator (STR) when the identified transfer function has pole-zero cancellation. In [5], a switching procedure is given that guarantees global stability of the closed-loop system. The procedure requires finding the Sylvester determinant (of an N x N matrix, with N the number of unknown parameters) to detect when near pole-zero cancellation occurs. We show that a Routh-like array can be used for this purpose with considerably less computational expense. The array also finds the nearly common factor.

Section III of the paper deals with finding the polynomial solution P(s) and Q(s) to the scalar Diophantine equation

$$A(s)P(s) + B(s)Q(s) = C(s) \ . \tag{1.1}$$

This equation is important in its own right and arises in many regulation and adaptive control problems [6]. Many techniques have been devised to solve equation (1), such as the one described in [7]. The algorithm presented here will yield a recursive solution to (1) by implementing a long division procedure that uses two Routh-like arrays.

II. The Cauchy Index Algorithm

There is given a linear nth-order, time-invariant, single-input/single-output (SISO), continuous-time system described by the transfer function

$$H(s) = \frac{B(s)}{A(s)} \tag{2.1}$$

where

$$A(s) = a_o s^n + a_1 s^{n-1} + \cdots + a_{n-1}s + a_n \ ,$$
$$B(s) = b_o s^m + b_1 s^{m-1} + \cdots + b_{m-1}s + b_m \ , \ m < n \ . \tag{2.2}$$

The Cauchy Index of H(s) is defined as the difference between the number of jumps of H(s) from $-\infty$ to $+\infty$ and the number of jumps from $+\infty$ to $-\infty$ as the argument s changes from $-\infty$ to $+\infty$. In [2], it was shown that the Cauchy index can be found by constructing a Sturm chain of polynomials and counting the number of variations of sign in the chain at $+\infty$ and $-\infty$, that is

$$\text{Cauchy Index of } H(s) \triangleq I_{-\infty}^{\infty} H(s) = V(-\infty) - V(\infty) \tag{2.3}$$

where V(a) is the number of variations of signs in the Sturm chain at s = a.

The Sturm chain can be constructed using long division as follows.

Let
$$r_{-1}(s) \triangleq A(s)$$
$$r_o(s) \triangleq B(s)$$

Then using the Euclidean algorithm, one gets

$$r_i(s) = r_{i+1}(s)q_{i+2}(s) + r_{i+2}(s) \quad ; \quad i = -1,\ldots,k-1 \tag{2.4}$$

where k is defined by $r_{k+2}(s) = 0$ so that

$$r_k(s) = r_{k+1}(s)q_{k+2}(s) \ .$$

The chain $r_{-1}(s)$, $r_o(s)$, $-r_1(s)$, ..., $-r_{k+1}(s)$ is a Sturm chain [2]. This chain can be alternatively constructed using a Routh-Hurwitz-like table as follows.

Let
$$a_i^1 = b_i \quad , \quad i = 0,1,\ldots,m$$
$$a_j^2 = a_j \quad , \quad j = 0,1,\ldots,n \tag{2.5a}$$

then construct Table 2.1, where for j = 3,...,n-m+1

$$a_i^j = \frac{a_{i+1}^{j-1} a_o^1 - a_o^{j-1} a_{i+1}^1}{a_o^1} \quad , \quad i = 0,\ldots,n-j+2 \ . \tag{2.5b}$$

then

$$a_i^{n-m+k} = \frac{a_o^{(n-m+k-2)} a_{i+1}^{(n-m+k-1)} - a_{i+1}^{(n-m+k-2)} a_o^{(n-m+k-1)}}{a_o^{(n-m+k-2)}} \tag{2.6}$$

or

$$a_i^j = \begin{cases} \dfrac{a_o^{j-2} a_{i+1}^{j-1} - a_{i+1}^{j-2} a_o^{j-1}}{a_o^{j-2}} & j > n-m+2 \text{ and } j-(n-m) \text{ odd} \\ & i = 0,1,\ldots,n-j+2 \quad (2.7a) \\ \dfrac{a_o^{j-2} a_{i+1}^{j-1} - a_{i+1}^{j-2} a_o^{j-1}}{a_o^{j-1}} & j > n-m+2 \text{ and } j-(n-m) \text{ even} \\ & i = 0,1,\ldots,n-j+2 \quad (2.7b) \end{cases}$$

Next, consider the coefficients

$$a_o^1,\ a_o^{n-m+3},\ \ldots,\ a_o^{n-m+(2\ell+1)},\ \ldots \tag{2.8}$$

That is, all first column elements (2.6) with odd k. Let

$N \triangleq$ # similar signs for consecutive elements

$K \triangleq$ # sign changes for consecutive elements

Then we have the following method of determining the Cauchy Index.

Theorem 2.1

$$I_{-\infty}^{\infty} H(s) = N-K \tag{2.9}$$

Proof:

$I_{-\infty}^{\infty} H(s) = V(-\infty) - V(\infty)$ as described in [2].

The Sturm chain is represented in our table as those polynomials with leading coefficients given in (2.8). The Sturm chain will change sign at $-\infty$ precisely when two consecutive coefficients (which will correspond to two polynomials of different degree parity, i.e., one has an odd degree, the other an even degree) have the same sign, that is, $V(-\infty) = N$.

Similarly at $+\infty$ two consecutive coefficients of different signs will cause the chain to have a sign change, so that $V(\infty) = K$. Therefore, $I_{-\infty}^{\infty} H(s) = N-K$. ●

The algorithm presented above can also be used to check for common or almost common factors, which we now demonstrate. As shown in [5], one can detect the presence of these factors by finding the Sylvester determinant. Our algorithm, however, requires less computation and has been found to be more sensitive.

Lemma 2.2

If A(s) and B(s) have some roots that are close together, that is, almost common factors, then the two polynomials can be written as

$$A(s) = A_1(s)X(s) \tag{2.10}$$

$$B(s) = B_1(s)\left[X(s) + \Delta X(s)\right] \tag{2.11}$$

where the norm of $\Delta X(s)$ is small, with the norm defined as follows.

$$\text{If} \quad X(s) = x_o s^{\ell} + x_1 s^{\ell-1} + \ldots + x_{\ell-1}s + x_{\ell} \tag{2.12}$$

$$\text{then} \quad |X(s)| = \sqrt{x_o^2 + x_1^2 + \ldots + x_{\ell-1}^2 + x_{\ell}^2}\ . \tag{2.13}$$

Proof:

Assume that s_o is a root of X(s) so that

$$X(s_o) = \sum_{i=0}^{\ell} x_i s_o^{\ell-i} = 0\ .$$

X(s) being a polynomial is continuous in s, that is: $\forall\ \delta > 0$ and $\forall$ s such that $|X(s) - X(s_o)| = |X(s)| < \delta$ there exists $\varepsilon > 0$ such that $|s - s_o| < \varepsilon$, or equivalently $\forall\ \varepsilon > 0$ such that $|s - s_o| > \varepsilon$, there exists δ such that $|X(s)| > \delta$. Next consider X(s) as a function of x_i for $i = 0,1,\ldots,\ell$ and let s' be fixed such that $|s' - s_o| > \varepsilon$. X(s') is continous with respect to each x_i, that is:

$$\forall\ x_i^* \text{ and } \forall\ \gamma > 0 \text{ such that } |X(s') - X^*(s')| < \gamma$$

there exists $\alpha_i > 0$ such that $|x_i - \overset{\bullet}{x}_i| < \alpha_i \forall\ i = 0,1,\ldots,n$, where

$$X^*(s) = \sum_{i=0}^{\ell} \overset{\bullet}{x}_i s^{\ell-i}\ .$$

In particular, let $\gamma = \delta/2$, then $|X(s') - X^*(s')| < \delta/2$. But $|X(s')| > \delta$, therefore, $|X^*(s')| > \delta/2 > 0$, that is s' is not a zero of $X^*(s)$. Therefore, no root of $X^*(s)$ can be found outside a small radius of the roots of X(s). Therefore, if the coefficients of X(s) change from x_i to $x_i^* = x_i + \delta_i$ where $|\delta_i| < \alpha_i$, the roots of X(s) can only move within a circle of radius $< \varepsilon$. This will guarantee that the roots of:

$$X^*(s) = X(s) + \Delta X(s)$$

and those of X(s) are close whenever $|\Delta X(s)|$ is small. Hence, when A(s) and B(s) have almost common factors, we can write

$$A(s) = A_1(s)X(s)$$

$$B(s) = B_1(s)\left[X(s) + \Delta X(s)\right]$$

where $\Delta X(s)$ is a polynomial of small norm.

The next result provides the test for almost common factors of two polynomials.

Theorem 2.3

If one row in Table 2.1 has a norm less than γ a small positive number, then A(s) and B(s) have an almost common factor.

Proof

The Euclidean division algorithm is (2.4), where r_{k+1} is the G.C.D. of A(s) and B(s). Now consider $A = A_1X$, $B = B_1(X + \Delta X)$, that is, the case of almost common factors. Then

$$\begin{aligned} A &= A_1X = Bq_1 + r_1 = B_1q_1X + B_1q_1\Delta X + r_1 \\ &= B_1Xq_1 + r_1' \end{aligned}$$

$$\begin{aligned} B &= B_1X + B_1\Delta X = r_1q_2 + r_2 = r_1'q_2 - B_1q_1\Delta Xq_2 + r_2 \\ B_1X &= r_1'q_2 + r_2 - B_1q_1q_2\Delta X - B_1\Delta X \\ &= r_1'q_2 + r_2 - (1 + q_1q_2)B_1\Delta X \\ &= r_1'q_2 + r_2' \\ r_1' &= r_2'q_3 + r_3 \\ &= r_2q_3 - q_3(1 + q_1q_2)B_1\Delta X + r_3 = r_2q_3 + r_3' \end{aligned}$$

$$\vdots$$

$$r_{k+1}' = r_{k+1} - \prod_{i=3}^{k+1} q_i(1 + q_1q_2)B_1\Delta X\ .$$

Suppose

$$\left|\prod_{i=3}^{k+1} q_i(1 + q_1q_2)B_1\Delta X\right| < \gamma\ .$$

where $|X|$ denotes the norm of a polynomial defined by (2.13). Then

$$\left|\prod_{i=3}^{k+1} q_i(1+q_1q_2)B_1\right|\cdot|\Delta X| < \left|\prod_{i=3}^{k+1} q_i(1+q_1q_2)B_1\Delta X\right| < \gamma$$

so that

$$|\Delta X| < \frac{\gamma}{\left|\prod_{i=3}^{k+1} q_i(1+q_1q_2)B_1\right|} < \frac{\gamma}{\prod_{i=3}^{k+1}|q_i|\cdot|1+q_1q_2|\cdot|B_1|}$$

$$< \frac{\gamma}{\prod_{i=1}^{k+1}|q_i|\cdot|B_1|}\,.$$

But

$$|B_1| > \beta \triangleq \max b_i^1 \quad\text{and}\quad |q_i| > \lambda \triangleq \max q_j^i\,,$$

where

$$B_1(s) = \sum_{i=0}^{r} b_i^1 s^{r-i} \quad\text{and}\quad q_i(s) = \sum_{j=0}^{r_i} q_j^i s^{r_i-j}$$

which yields

$$|\Delta X| < \frac{\gamma}{\beta\,\lambda^{n+1}} \triangleq \gamma_1\,.$$

Therefore, if one row in our table has a norm less than γ, there exists an almost common factor between $A(s)$ and $B(s)$.

It should be noted that, if any row has a small norm, then the almost-common factor is given by the polynomial two rows above.

III. The Diophantine Equation Algorithm

A modified Routh-table can also be used to solve the Diophantine equation.

Given the polynomials $A(s)$, $B(s)$ described in Section II and a polynomial

$$C(s) = c_o s^p + c_1 s^{p-1} + c_2 s^{p-2} + \dots c_{p-1}s + c_p\,, \qquad (3.1)$$

the Diophantine equation is

$$A(s)P(s) + B(s)Q(s) = C(s)\,. \qquad (3.2)$$

The object is to find the polynomials $P(s)$ and $Q(s)$.

Let $Y_{-1} = A$, $Y_o = B$, then the divisions

$$Y_i = Y_{i+1}X_{i+2} + Y_{i+2} \qquad i = -1,\dots,k-2 \qquad (3.3)$$

define X_{i+2} and Y_{i+2}, and k is defined by $Y_{k+1} = 0$ so that

$$Y_{k-1} = Y_k X_{k+1}\,. \qquad (3.4)$$

Y_k is the highest common denominator of A and B and there exists a solution to (3.2) if and only if $C = C_kY_k$ for some polynomial C_k.

Let $P_o = 1$, $P_1 = -X_1$, $Q_o = 0$, and $Q_1 = 1$, and define P_j, Q_j for $j > 1$ by

$$P_{i+2} = P_i - X_{i+2}P_{i+1} \qquad (3.5a)$$

$$Q_{i+2} = Q_i - X_{i+2}Q_{i+1} \qquad i = 0,1,\dots,k-2\,. \qquad (3.5b)$$

Then it may easily be shown that the solution to (3.2) is given by

$$P = C_kP_k\,,\ Q = C_kQ_k\,. \qquad (3.6)$$

These machinations may be implemented in a modified Routh algorithm. The algorithm consists of two tables. The first is identical to Table 2.1 and generates the Y_i's and X_i's. The Y_i's are the r_i's of the previous section and the X_i's are given as follows:

$$X_1 = \frac{a_o}{b_o}s^{n-m} + \frac{c_o}{b_o}s^{n-m-1} + \dots + (\ )s^o$$

or in our new notation:

$$X_1 = \frac{a_o^2}{a_o^1}s^{n-m} + \frac{a_o^3}{a_o^1}s^{n-m-1} + \dots + \frac{a_o^{n+m+1}}{a_o^1}s^o$$

$$X_2 = \frac{a_o^1}{a_o^{n-m+3}}s + \frac{a_o^{n-m+4}}{a_o^{n-m+3}}$$

$$X_3 = \frac{a_o^{n-m+3}}{a_o^{n-m+5}}s + \frac{a_o^{n-m+6}}{a_o^{n-m+5}}$$

$$\vdots$$

$$X_i = \frac{a_o^{n-m+2i-3}}{a_o^{n-m+2i-1}}s + \frac{a_o^{n-m+2i}}{a_o^{n-m+2i-1}}\,.$$

Let $\quad -X_i(s) = x_i^1 s + x_i^2$

$$P_i(s) = P_i^1 s + P_i^2 s^2 + \dots + P_i^{n_i}$$

where n_i is the degree of $P_i(s)$. Therefore, we arrive at Table 3.1 which generates the polynomials P_i. This table is filled in as follows.

$-X_{i+1}$	x_{i+1}^1	x_{i+1}^2		
P_i	$P_{i-1}^1x_i^1$	$P_{i-1}^2x_i^1 + P_{i-1}^1x_i^2$	$P_{i-1}^3x_i^1 + P_{i-1}^2x_i^2$	$\dots$

Table 3.2 will produce P_k as its last entry, therefore

$$P = C_kP_k \text{ gives } P\,.$$

A similar procedure will give Q with $Q_o = 0$ and $Q_1 = 1$.

Conclusions

Algorithms for finding the Cauchy index and solving the Diophantine equation were introduced. The algorithms have a simple structure and exploit the common aspect of the two problems, which is the implementation of the Euclidean division. The first algorithm also gives a test for almost common factors of two polynomials which is simpler than previously given tests.

These algorithms should be useful in identification problems for one can check for structural changes using the Cauchy index information, and in control problems for one can solve the Diophantine equation and check for almost common factors. The ability of the first algorithm to find (not just detect) almost common factors makes it potentially useful as an alternative to [8] for the adaptive control of nonminimum-phase systems.

References

1. Roger W. Brockett, "Some Geometric Questions in the Theory of Linear Systems," IEEE Trans. Automat. Control, Vol. AC-21, No. 4, pp. 449-454, Aug. 1976.

2. F. R. Gantmacher, The Theory of Matrices, pp. 172-185, Chelsea Publishing Company.
3. B. D. O. Anderson, "On the Computation of the Cauchy Index," Quart. Appl. Math., Vol. 29, pp. 577-582, Jan. 1972.
4. K. V. Fernando and H. Nicholson, "On the Cauchy Index of Linear Systems," IEEE Trans. Automat. Control, Vol. AC-28, No. 2, pp. 222-224, Feb. 1983.
5. B. D. O. Anderson and R. M. Johnstone, "Global Adaptive Pole Positioning," IEEE Trans. Automat. Control, Vol. AC-30, No. 1, pp. 11-22, Jan. 1985.
6. K. J. Aström, "Self Tuning Regulators - Design Principles and Applications," in Applications of Adaptive Control, pp. 1-68, Academic Press, 1980.
7. V. Kučera, Discrete Linear Control: The Polynomial Equation Approach, pp. 148-151, John Wiley & Sons, 1979.
8. K. J. Aström, "Direct Methods for Nonminimum Phase Systems," Proc. 19th IEEE Conf. Dec. and Control, pp. 611-615, Albuquerque, NM, December 1980.

TABLE 2.1

s^m	a_o^1	a_1^1	$a_2^1 \;. \;. \;. \; a_m^1$
s^n	a_o^2	a_1^2	$a_2^2 \;. \;. \;. \; a_m^2 \;. \;. \;. \; a_m^2$
s^{n-1}	a_o^3	a_1^3	$a_2^3 \;. \;. \;. \;. \;. \; a_{n-1}^3$
s^{n-2}	a_o^4	a_1^4	
s^{n-3}	a_o^5	a_1^5	
$\vdots$	$\vdots$	$\vdots$	
s^m	a_o^{n-m+2}	. . . .	
s^{m-1}	a_o^{n-m+3}	. . . .	
s^{m-1}	a_o^{n-m+4}	. . . .	
s^{m-2}	a_o^{n-m+5}	. . . .	
$\vdots$			
s^1			
s^1			
s^o	a_o^{n+m+1}		

TABLE 3.1

P_o	1			
$-X_2$	$-\frac{a_o^1}{a_o^{n-m+3}}$	$-\frac{a_o^{n-m+4}}{a_o^{n-m+3}}$		
$(-X_1) = P_1$	$-\frac{a_o^2}{a_o^1}$	$-\frac{a_o^3}{a_o^1}$	$\cdots$	$-\frac{a_o^{n+m+1}}{a_o^1}$
$-X_3$	$-\frac{a_o^{n-m+3}}{a_o^{n-m+3}}$	$-\frac{a_o^{n-m+6}}{a_o^{n-m+5}}$		
P_2	$\frac{a_o^2}{a_o^{n-m+3}}$	$\frac{a_o^3}{a_o^{n-m+3}} + \frac{a_o^2}{a_o^1} \cdot \frac{a_o^{n-m+4}}{a_o^{n-m+3}}$		
$-X_4$	$-\frac{a_o^{n-m+5}}{a_o^{n-m+7}}$	$-\frac{a_o^{n-m+8}}{a_o^{n-m+7}}$		
$\vdots$				

TABLE 3.2

P_o	1			
$-X_2$	x_2^1	x_2^2		
P_1	p_1^1	p_1^2	p_1^3	
$-X_3$	x_3^1	x_3^2		
P_2	$\underbrace{p_1^1x_2^1}_{p_2^1}$	$\underbrace{p_1^2x_2^1 + p_1^1x_2^2}_{p_2^2}$	$\underbrace{p_1^3x_2^1 + p_1^2x_2^2}_{p_2^3}$	$\underbrace{p_1^4x_2^1 + p_1^3x_2^1}_{p_2^4}$
$-X_4$	x_4^1	x_4^2		
P_3	$p_2^1x_3^1$	$p_2^2x_3^1 + p_2^1x_3^2$	$p_2^3x_3^1 + p_2^2x_3^2$	$\cdots$
$\vdots$				