12-1-1986

A Routh-like algorithm in system identification and adaptive control

Chaouki T. Abdallah

F. L. Lewis

B. L. Stevens

Follow this and additional works at: http://digitalrepository.unm.edu/ece_fsp

Recommended Citation


This Article is brought to you for free and open access by the Engineering Publications at UNM Digital Repository. It has been accepted for inclusion in Electrical & Computer Engineering Faculty Publications by an authorized administrator of UNM Digital Repository. For more information, please contact disc@unm.edu.
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 nearly 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

\[
A(s)P(s) + B(s)Q(s) = C(s) \quad (1.1)
\]

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

\[
H(s) = \frac{B(s)}{A(s)} \quad (2.1)
\]

where

\[
A(s) = a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0,
\]

\[
B(s) = b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0, \quad m < n.
\]

The Cauchy Index of \( H(s) \) is defined as the difference between the number of jumps of \( H(s) \) from \(-\) to \(+\) and the number of jumps from \(+\) to \(-\) as the argument \( s \) changes from \(-\infty\) to \(+\infty\). In [2], it is 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 \( +\) and \( -\), that is

Cauchy Index of \( H(s) \) is \( \sum V(s) = V(+) - V(-) \) (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

\[
r_i(s) = A(s), \quad r_0(s) \equiv B(s)
\]

Then using the Euclidian algorithm, one gets

\[
r_i(s) = r_{i+1}(s)q_{i+2}(s) + r_{i+2}(s), \quad i = -1, \ldots, k-1
\]

where \( k \) is defined by \( r_k(s) = 0 \) so that

\[
r_k(s) = r_{k+2}(s)q_{k+2}(s).
\]

The chain \( r_{-1}(s), r_0(s), -r_1(s), \ldots, -r_{k+1}(s) \) is a Sturm chain [2]. This chain can be alternatively constructed using a Routh-Hurwitz-like table as follows.

Let

\[
a_j = b_i, \quad i = 0, 1, \ldots, m
\]

\[
a_j = a_i, \quad j = 0, 1, \ldots, n
\]

then construct Table 2.1, where for \( j = 3, \ldots, n-m+1 \)

\[
a_j = \frac{a_j}{a_0} a_i, \quad i = 0, \ldots, n-j+2.
\]

then

\[
A_{n-m+k} = \frac{a_0 (n-m+k-2) - a_{n-m+k-2} a_0}{a_0 (n-m+k-2)}
\]

or

\[
A_{n-m+k} = \frac{a_0 (n-m+k-2) - a_{n-m+k-2} a_0}{a_0 (n-m+k-2)}
\]

\[
A_{n-m+k} = \frac{a_0 (n-m+k-2) - a_{n-m+k-2} a_0}{a_0 (n-m+k-2)}
\]
where the norm of $AX(s)$ is small, with the norm defined as $\| . \|_N$. Similarly at $-\infty$, $H(s)$ has an odd degree, the other an even degree) have the same sign, that is: $\text{V}(-m) = \text{V}(m)$ as described in (2.4).

Let $N = \#$ similar signs for consecutive elements

$K = \#$ sign changes for consecutive elements

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

**Theorem 2.1**

$$I_{\text{Cauchy}}(s) = N - K$$

**Proof:**

$$i = 0, 1, ..., n$$

The Sturm chain is represented in our table as those polynomials with leading coefficients given in (2.8). Similarly at $-\infty$, 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, $V(-m) = N$. Therefore, $I_{\text{Cauchy}}(s) = N - K$.

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

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

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

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

If

$$X(s) = x_0 s^k + x_1 s^{k-1} + ... + x_{k-1} s + x_k$$

then

$$|X(s)| = \sqrt{x_0^2 + x_1^2 + ... + x_{k-1}^2 + x_k^2}$$

**Proof:**

Assume that $s_0$ is a root of $X(s)$ so that

$$X(s_0) = \sum_{i=0}^k x_i s_0^{k-i} = 0.$$

Then $X(s)$ being a polynomial is continuous in $s$, that is: there exists $\delta > 0$ such that $|s - s_0| < \delta$, there exists $\epsilon > 0$ such that $|s - s_0| < \epsilon$, or equiva-
where \(|X|\) denotes the norm of a polynomial defined by
\[(2.13)\]
Then
\[
\sum_{i=3}^{k+1} q_i (1 + q_i q_2) B_i |AX| < \sum_{i=3}^{k+1} |q_i (1 + q_i q_2) B_i AX| < \gamma
\]
so that
\[
|AX| < \gamma \frac{\xi}{\lambda} \gamma_{k+1}. 
\]
Therefore, if one row in our table has a norm less than \(\gamma\), 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
\[
C(s) = c_n s^n + c_{n-1} s^{n-1} + c_{n-2} s^{n-2} + \ldots + c_0 s + c_p \quad (3.1)
\]
the Diophantine equation is
\[
A(s)P(s) + B(s)Q(s) = C(s). \quad (3.2)
\]
The object is to find the polynomials \(P(s)\) and \(Q(s)\).

Let \(Y_{-1} = X, Y_0 = B\), then the divisions
\[
Y_1 = Y_{i+1} X_{i+2} + Y_{i+2} \quad i = -1, \ldots, k-2 \quad (3.3)
\]
define \(X_{i+2}\) and \(Y_{i+2}\), and \(k\) is defined by \(Y_{k+1} = 0\) so that
\[
Y_{k-1} = Y_k X_{k+1}. \quad (3.4)
\]
\(X_k\) is the highest common denominator of \(A\) and \(B\) and there exists a solution to (3.2) if and only if \(C = C_k X_k\) for some polynomial \(C_k\).

Let \(P_0 = 1, P_1 = -X_1, Q_0 = 0, \) and \(Q_1 = 1,\) and define \(P_j, Q_j\) for \(j > 1\) by
\[
P_{i+2} = P_i - X_{i+2} P_{i+1} \quad (3.5a)
\]
\[
Q_{i+2} = Q_i - X_{i+2} Q_{i+1} \quad i = 0, 1, \ldots, k-2. \quad (3.5b)
\]
Then it may easily be shown that the solution to (3.2) is given by
\[
P = C_k P_k, \quad Q = C_k Q_k. \quad (3.6)
\]
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 \(X_i\)’s and \(Y_i\)’s. The \(Y_i\)’s are the \(r_i\)’s of the previous section and the \(X_i\)’s are given as follows:
\[
X_1 = \begin{pmatrix} \frac{a_0}{b_0} s^{n-m} + \frac{a_0}{b_0} s^{n-m-1} + \ldots \end{pmatrix} e^0
\]
or in our new notation:
\[
\begin{align*}
X_1 &= \frac{a_1}{a_0} s^{n-m} + \frac{a_0}{a_1} s^{n-m-1} + \ldots + \frac{a_n}{a_0} s^0 \\
X_2 &= \frac{a_1}{a_0} s^{n-m+2} + \frac{a_0}{a_1} s^{n-m+3} + \ldots + \frac{a_n}{a_0} s^{n-m+6} \\
X_j &= \frac{a_1}{a_0} s^{n-m+j} + \frac{a_0}{a_1} s^{n-m+j+1} \\
X_k &= -X_1 X_{k+1} + 2 X_{k+1}^2
\end{align*}
\]
where \(n_1\) is the degree of \(P_j(s)\). Therefore, we arrive at Table 3.1 which generates the polynomials \(P_j\). This table is filled in as follows.

\[
\begin{array}{c|cccc}
0 & 1 & 1 & 2 & 1 \hline
1 & 1 & 1 & 2 & 1
\end{array}
\]
Table 3.2 will produce \(P_k\) as its last entry, therefore \(P = C_k P_k\) gives \(P\).

A similar procedure will give \(Q\) with \(Q_0 = 0\) 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**

TABLE 2.1

<table>
<thead>
<tr>
<th>( s^m )</th>
<th>( a_1 )</th>
<th>( a_2 )</th>
<th>( \ldots )</th>
<th>( a_m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s^n )</td>
<td>( a_2 )</td>
<td>( a_1 )</td>
<td>( a_2 )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( s^{n-1} )</td>
<td>( a_3 )</td>
<td>( a_2 )</td>
<td>( a_3 )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( s^{n-2} )</td>
<td>( a_4 )</td>
<td>( a_3 )</td>
<td>( a_4 )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( s^{n-m+2} )</td>
<td>( a_{n-m+2} )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( s^{n-m+1} )</td>
<td>( a_{n-m+1} )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( s^{n-m} )</td>
<td>( a_{n-m} )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( s^{n-m+1} )</td>
<td>( a_{n-m+1} )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
</tbody>
</table>

TABLE 3.1

\[
\begin{array}{ccc}
\mathbf{X}_2 & \frac{a_1}{a_0} & \frac{a_{n-m+4}}{a_0} \\
\mathbf{X}_3 & \frac{a_2}{a_0} & \frac{a_{n-m+5}}{a_0} \\
\mathbf{X}_4 & \frac{a_3}{a_0} & \frac{a_{n-m+6}}{a_0} \\
\end{array}
\]

TABLE 3.2

\[
\begin{array}{cccc}
P_0 & \frac{1}{1} & \frac{a_1}{a_0} & \frac{a_{n-m+4}}{a_0} \\
P_1 & \frac{x_1}{p_1} & \frac{x_2}{p_1} & \frac{x_3}{p_1} \\
P_2 & \frac{x_1^2}{p_2} & \frac{x_2^2}{p_2} & \frac{x_3^2}{p_2} \\
P_3 & \frac{x_1^3}{p_3} & \frac{x_2^3}{p_3} & \frac{x_3^3}{p_3} \\
\end{array}
\]