AN ABSTRACT OF THE THESIS OF

John Ries Holroyd for the Master of Science in Mathematics
(Name) (Degree) (Major)

Date thesis is presented Nov. 8, 1962

Title ALGORITHMS FOR THE SOLUTION OF TWO ALGEBRAIC EQUATIONS IN TWO UNKNOWNS

Abstract approved [redacted]
(Major Professor)

The author develops a new recursive procedure for the evaluation of Sylvester's eliminant. The algorithm is written in the ALGOL 60 language.

ALGORITHMS FOR THE SOLUTION OF TWO ALGEBRAIC EQUATIONS IN TWO UNKNOWNS

by

JOHN RIES HOLROYD

A THESIS

submitted to

OREGON STATE UNIVERSITY

in partial fulfillment of
the requirements for the
degree of

MASTER OF SCIENCE

June 1963

APPROVED:

[redacted]

Professor of Mathematics

In Charge of Major

[redacted]

Head of Department of Mathematics

[redacted]

Chairman of School Graduate Committee

[redacted]

Dean of Graduate School

Date thesis is presented Nov. 8, 1962

Typed by Carol Baker

ACKNOWLEDGMENT

The author would like to thank Dr. Goheen for his help in preparing this paper.

TABLE OF CONTENTS

ALGORITHMS FOR THE SOLUTION OF TWO ALGEBRAIC EQUATIONS IN TWO UNKNOWNS

CHAPTER I

INTRODUCTION

Algorithms for machine solution of systems of equations of degree one have been in use for many years. In this paper the author develops algorithms for the machine solution of two equations in two unknowns of degrees n and s, respectively, in the variable x with coefficients which are polynomials in y.

Consider two polynomial equations in x and y of the form

$$f(x, y) = \sum_{i=0}^{n} a_i(y)x^i = 0$$

$$g(x, y) = \sum_{i=0}^{s} b_i(y)x^i = 0$$

where $a_i(y)$ and $b_i(y)$ are polynomials in y. We may separate variables by use of Sylvester's method of elimination which yields the following determinantal equation:

$$A=\left|\begin{array}{l}
\left.\begin{array}{l}
a_n \quad a_{n-1} \; a_{n-2} \; \cdots \quad a_1 \; a_0 \quad 0 \; \cdots \\
0 \quad a_n \quad a_{n-1} \; a_{n-2} \; \cdots \quad a_1 \quad a_0 \quad 0 \; \cdots \\
\qquad\qquad\qquad\qquad \vdots \\
\qquad\quad 0 \quad a_n \quad a_{n-1} \; a_{n-2} \; \cdots \quad a_1 \quad a_0 \quad 0 \\
\qquad\qquad\; 0 \quad a_n \quad a_{n-1} \, a_{n-2} \; \cdots \quad a_1 \quad a_0
\end{array}\right\} s \text{ rows} \\
\left.\begin{array}{l}
b_s \quad b_{s-1} \; \cdots \quad b_1 \quad b_0 \quad 0 \\
0 \quad b_s \quad b_{s-1} \; \cdots \quad b_1 \quad b_0 \quad 0 \\
\qquad\qquad\qquad\qquad \vdots \\
\qquad\qquad\qquad 0 \quad b_s \; b_{s-1} \; \cdots \quad b_1 \quad b_0
\end{array}\right\} n \text{ rows}
\end{array}\right. =0. \qquad (1)$$

This may be written in the form

$$\sum_{i=0}^{r} c_i y^i = 0, \qquad (2)$$

where

$r \leqslant s$ ($\max_i$ degree a_i) + n ($\max_i$ degree b_j). The equation (1) has a set of solutions $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_r$, provided that (1) is not an identity. The case that (1) is an identity will not be considered in this paper.

The following theorem may be proven.

<u>I Theorem:</u> If not both $a_n(\alpha_i)$ and $b_s(\alpha_i)$ are zero, then $f(x, \alpha_i)$ and $g(x, \alpha_i)$ have a common root.

This is a well-known result from the standard theory of equations.

It is the purpose of this paper to show a recursive procedure for the evaluation of the determinant A.

For convenience we may write the right hand side of (1) as

$$\begin{vmatrix}
a_n & a_{n-1} a_{n-2} & \cdots & a_{m+1} a_m & 0 & \cdots & 0 \\
0 & a_n & a_{n-1} a_{n-2} \cdots & & a_{m+1} a_m & 0 \cdots & 0 \\
 & & & \vdots & & & \\
0 & & 0 & a_n & a_{n-1} a_{n-2} & & a_{m+1} a_m \\
b_s & b_{s-1} & \cdots & b_{k+1} b_k & 0 & \cdots & 0 \\
0 & b_s & b_{s-1} & \cdots & b_{k+1} b_k & 0 \cdots & 0 \\
 & & & \vdots & & & \\
0 & \cdots & 0 & b_s & b_{s-1} & \cdots & b_{k+1} b_k
\end{vmatrix} \tag{3}$$

Concerning the determinant (3) we may state the following theorems:

II Theorem: The eliminant (3) may be reduced recursively in such a manner that the eliminant is equal to the product of one of the elements: $a_n(y)$, $b_k(y)$, $a_m(y)$ or $b_s(y)$ and an eliminant of lower order whose elements are rational functions of the elements of the original eliminant and hence rational functions of y provided that n-m and s-k are both greater than one.

III Theorem: In case n-m or s-k is one, the recursion is defined and the eliminant is the product of a rational function and diagonal determinant whose non-zero elements are all the same rational function.

The proofs of theortems II and III are specific as to the means of reduction and the theorems will be restated as theorems IV and V in Chapter II incorporating the means of reduction in their statements. Algorithms based on theorems IV and V are given in Chapter III.

CHAPTER II

RECURSION FOR SYLVESTER'S DETERMINANT

Consider the following determinant:

$$\begin{array}{c|cccccc|}
 & a_n & a_{n-1}a_{n-2}\cdots & & a_{m+1}a_m & 0\cdots & 0 \\
 & 0 & a_n & a_{n-1}a_{n-2}\cdots & & a_{m+1}a_m\ 0\cdots & 0 \\
s-k & & & \cdot & & & \\
\text{rows} & & & \cdot & & & \\
 & & & & \cdot & & \\
 & 0 & & 0 & a_n & a_{n-1}a_{n-2} & a_{m+1}a_m \\
 & b_s & b_{s-1}\cdots & & b_{k+1}b_k & 0\ \cdots & 0 \\
 & 0 & b_s & b_{s-1}\cdots & & b_{k+1}b_k\ \cdots & 0 \\
 & & & \cdot & & & \\
n-m & & & \cdot & & & \\
\text{rows} & & & \cdot & & & \\
 & 0 & \cdots & 0 & b_s & b_{s-1}\cdots & b_{k+1}b_k
\end{array} . \qquad (4)$$

If $m = 0$ and $k = 0$, then the determinant (4) is Sylvester's eliminant as described in part one. Let m and k be arbitrary integers such that

$$0 \leqslant m < n \qquad \text{and} \qquad 0 \leqslant k < s.$$

Let the following equalities hold:

$$u = n, \quad v = s, \quad p = m, \quad q = k.$$

One and only one of the following elementary transformations is defined on the determinant (4).

Case I: $a_u \not\equiv 0$ and $u - p \leqslant v - q$. Let

$$b'_{v+i-u} = b_{v+i-u} - \frac{a_i b_v}{a_u} .$$

In the determinant (4), replace b_{v+i-u} by b'_{v+i-u} for $i = p, p + 1, p + 2, \ldots, u$ in every row containing b_{v+i-u}. Expand the determinant (4) by the element a_u in the left most column and redefine the determinant (4) to be the remaining determinant. Replace the value of v by the value of $v - 1$.

Case II: $a_u \equiv 0$, $u - p \leqslant v - q$, and $b_v \not\equiv 0$. Expand the determinant (4) by the element b_v in the left most column and redefine the determinant (4) to be the remaining determinant. Replace the value of u by the value of $u - 1$.

Case III: $b_q \not\equiv 0$ and $u - p > v - q$. Let

$$a'_{p+i-q} = a_{p+i-q} - \frac{b_i a_p}{b_q} .$$

In the determinant (4) replace a_{p+i-q} by a'_{p+i-q} for $i = q, q + 1, \ldots, v$ in every row containing a_{p+i-q}. Expand the determinant (4) by the element b_q in the right most column and redefine (4) to be the remaining determinant. Replace the value of p by the value of p + 1.

Case IV. $b_q \equiv 0$, $u-p > v-q$, and $a_p \not\equiv 0$. Expand by the element a_p in the right most column and redefine the determinant (4) to be the remaining determinant. Replace the value of q by the value of q + 1.

Case V: If both a_u and b_v (or, a_p and b_q) are identically zero, then the algorithms developed in this paper do not apply, this yielding the excluded case of chapter one.

IV Theorem: If n - m and s - k are greater than one, then the transformed determinant defined by the operations I through IV is of the same form as the determinant (4). Thus the determinant (4) is evaluated as the product of a rational function of y and a transformed determinant which is also an eliminant.

Proof: The determinant (4) has two sets of elements a_i ($i=m, m+1, \ldots, n$) and b_j ($j=k, k+1, \ldots, s$), which are arranged in the form of Sylvester's eliminant and satisfy the relations:

1) the number of rows containing a_i is equal to s - k.

2) the number of rows containing b_j is equal to n - m.

I: Assume $n - m \leqslant s - k$ and $a_n \not\equiv 0$. Case I applies. If b_{s+i-n} is replaced by b'_{s+i-n} in the determinant (4), the following determinant results:

$$\begin{vmatrix}
a_n & a_{n-1} & \cdots & a_m & & & & \\
0 & a_n & a_{n-1} & \cdots & a_m & & & \\
 & & & & & & & \\
 & & & 0 & a_n & a_{n-1} & \cdots & a_m \\
0 & b'_{s-1} & \cdots & b'_{k+1} & b'_k & & & \\
0 & 0 & b'_{s-1} & \cdots & b'_{k+1} & b'_k & & \\
 & & & & & & & \\
 & & 0 & 0 & b'_{s-1} & \cdots & b'_{k+1} & b'_k
\end{vmatrix}.$$

The minor of the element a_n in the left most column is:

$$\begin{vmatrix} a_n & a_{n-1}\cdot\cdot\cdot & a_m & & & \\ 0 & a_n & a_{n-1}\cdot\cdot\cdot & a_m & & \\ & & \vdots & & & \\ & & 0 & a_n & a_{n-1}\cdot\cdot\cdot & a_m \\ b'_{s-1}\cdot\cdot\cdot & & b'_{k+1} & b'_k & & \\ & & \vdots & & & \\ & b'_{s-1} & & & b'_{k+1} & b'_k \end{vmatrix} . \qquad (5)$$

The determinant (5) has two sets of elements a_i ($i=m,\ m+1,\ \ldots,n$) and b'_j ($j=k,\ k+1,\ \ldots\ ,\ s-1$) which are arranged in the form of Sylvester's eliminant.

The number of rows containing a_i($i= m, m+1,\ \ldots\ , n$) is reduced by one. The number of rows containing b'_j($j=k, k+1,\ \ldots\ , s-1$) is unchanged. Thus we have the relations:

1) the number of rows containing a_i equals $s-1-k$.

2) the number of rows containing b'_j equals $n-m$.

Thus (5) has the form of (4).

II: Assume $n-m \leq s-k$, $a_n \equiv 0$ and $b_s \not\equiv 0$. Case II applies. Expanding (4) by the element b_s in the left most column the following determinant results:

$$\begin{vmatrix} a_{n-1} & a_{n-2} & \cdots & a_m & & & & \\ 0 & a_{n-1} & a_{n-2} & \cdots & a_m & & & \\ & & & & & & & \\ & & & & a_{n-1} & a_{n-2} & \cdots & a_m \\ b_s & b_{s-1} & \cdots & b_{k+1} & b_k & & & \\ 0 & b_s & b_{s-1} & \cdots & b_{k+1} & b_k & & \\ & & & & & & & \\ & & 0 & b_s & b_{s-1} & \cdots & b_{k+1} & b_k \end{vmatrix} \qquad . \tag{6}$$

The number of rows containing a_i ($i = n-1, n-2, \ldots, m$) is unchanged and is equal to $s-k$. The number of rows containing b_j ($j=k, k+1, \ldots, s$) has been reduced by one and is therefore equal to $n-1-m$. The determinant (6) has the same form as (4).

III: Assume $n - m > s - k$ and $b_k \not\equiv 0$. Case III applies. If a_{m+i-k} is replaced by a'_{m+i-k} the following determinant results:

$$\begin{vmatrix}
a'_n & a'_{n-1} & \cdots & a'_{m+1} & 0 & & & & \\
0 & a'_n & a'_{n-1} & \cdots & a'_{m+1} & 0 & & & \\
 & & & & \ddots & & & & \\
 & & & & a'_n & a'_{n-1} & \cdots & a'_{m+1} & 0 \\
b_s & b_{s-1} & \cdots & b_{k+1} & b_k & & & & \\
0 & b_s & b_{s-1} & \cdots & b_{k+1} & b_k & & & \\
 & & & \ddots & & & & & \\
 & & & b_s & b_{s-1} & \cdots & b_{k+1} & b_k &
\end{vmatrix} . \qquad (7)$$

Expanding (7) by the element b_k in the right most column the following determinant results:

$$\begin{vmatrix}
a'_n & a'_{n-1} & \cdots & a'_{m+1} & & & & \\
0 & a'_n & a'_{n-1} & \cdots & a'_{m+1} & & & \\
 & & & & a'_n & a'_{n-1} & \cdots & a'_{m+1} \\
b_s & b_{s-1} & \cdots & b_{k+1} & b_k & & & \\
0 & b_s & b_{s-1} & \cdots & b_{k+1} & b_k & & \\
 & & & b_s & b_{s-1} & \cdots & b_{k+1} & b_k
\end{vmatrix} . \qquad (8)$$

This determinant has two sets of elements, a_i' $(i=m+1, \ldots, n)$ and $b_j (j=k, \ldots, s)$ arranged in the form of Sylvester's eliminant. The number of rows containing $a_i'(i=m+1, m+2, \ldots, n)$ is equal to $s-k$. The number of rows containing b_j has been reduced by one. Therefore the number of rows containing b_j is equal to $n-m-1$. The determinant (8) is of the same form as the determinant (4).

IV: Assume $n-m > s-k$, $b_k \equiv 0$, and $a_m \not\equiv 0$. Case IV applies. Expanding by the element a_m in the right most column of the determinant (4) the following determinant results:

$$\begin{vmatrix}
a_n & a_{n-1} & \cdots & a_m & & & \\
0 & a_n & a_{n-1} & \cdots & a_m & & \\
 & & & & & & \\
 & & & a_n & a_{n-1} & \cdots & a_m \\
b'_s & b'_{s-1} & \cdots & b'_{k+2} & b'_{k+1} & & \\
0 & b'_s & b'_{s-1} & \cdots & b'_{k=2} & b'_{k+1} & \\
 & & & b'_s & b'_{s-1} & \cdots & b'_{k+2} \; b'_{k+1}
\end{vmatrix} . \qquad (9)$$

This determinant has two sets of elements a_i (i=m, m+l, . . . , n) and b_j (j=k+l, k + 2, . . . , s) arranged in the form of Sylvester's eliminant. The number of rows containing b_j is equal to n-m. The number of rows containing a_i is reduced by one. Therefore the number of rows containing a_i is equal to s - k - l. The determinant (9) is of the same form as (4)

This completes the proof of theorem IV. The following examples are included to illustrate each of the four cases and the application of the procedure to an actual problem.

Example I:

$$\begin{vmatrix} a_2 & a_1 & a_0 & 0 \\ 0 & a_2 & a_1 & a_0 \\ b_2 & b_1 & b_0 & 0 \\ 0 & b_2 & b_1 & b_0 \end{vmatrix} =D .$$

Case I applies giving the following expansion:

$$D=a_2 \begin{vmatrix} a_2 & a_1 & a_0 \\ b_1 - \frac{b_2a_1}{a_2} & b_0 - \frac{b_2a_0}{a_2} & 0 \\ 0 & b_1 - \frac{b_2a_1}{a_2} & b_0 - \frac{b_2a_0}{a_2} \end{vmatrix} .$$

Example II:

$$\begin{vmatrix} 0 & a_1 & a_0 & 0 \\ 0 & 0 & a_1 & a_0 \\ b_2 & b_1 & b_0 & 0 \\ 0 & b_2 & b_1 & b_0 \end{vmatrix} .$$

Case II applies giving the following expansion:

$$D = b_2 \begin{vmatrix} a_1 & a_0 & 0 \\ 0 & a_1 & a_0 \\ b_2 & b_1 & b_0 \end{vmatrix} .$$

Example III:

$$D = \begin{vmatrix} a_3 & a_2 & a_1 & a_0 \\ b_1 & b_0 & 0 & 0 \\ 0 & b_1 & b_0 & 0 \\ 0 & 0 & b_1 & b_0 \end{vmatrix} .$$

Case III applies giving the following expansion:

$$D = b_0 \begin{vmatrix} a_3 & a_2 & a_1 - \dfrac{b_1 a_0}{b_0} \\ b_1 & b_0 & 0 \\ 0 & b_1 & b_0 \end{vmatrix} .$$

Example IV:

$$D = \begin{vmatrix} a_3 & a_2 & a_1 & a_0 \\ b_1 & 0 & 0 & 0 \\ 0 & b_1 & 0 & 0 \\ 0 & 0 & b_1 & 0 \end{vmatrix} .$$

Case IV applies giving the following expansion:

$$D = a_0 \begin{vmatrix} b_1 & 0 & 0 \\ 0 & b_1 & 0 \\ 0 & 0 & b_1 \end{vmatrix} .$$

Example V:

Consider the set of equations in two unknowns:

$$\begin{aligned} (y+1)x^2 + yx + 1 &= 0 \\ yx^2 + (y+2)x + y &= 0 . \end{aligned}$$

$$\begin{vmatrix} y+1 & y & 1 & 0 \\ 0 & y+1 & y & 1 \\ y & y+2 & y & 0 \\ 0 & y & y+2 & y \end{vmatrix} =0.$$

$$\begin{vmatrix} y+1 & y & 1 & 0 \\ 0 & y+1 & y & 1 \\ 0 & y+2-\frac{y^2}{y+1} & y-\frac{y}{y+1} & 0 \\ 0 & 0 & y+2-\frac{y^2}{y+1} & y-\frac{y}{y+1} \end{vmatrix}$$

$$=y+1 \begin{vmatrix} y+1 & y & 1 \\ \frac{3y+2}{y+1} & \frac{y^2}{y+1} & 0 \\ 0 & \frac{3y+2}{y+1} & \frac{y^2}{y+1} \end{vmatrix}$$

$$=y+1 \begin{vmatrix} y+1 & y-\frac{3y+2}{y^2} & 0 \\ \frac{3y+2}{y+1} & \frac{y^2}{y+1} & 0 \\ 0 & \frac{3y+2}{y+1} & \frac{y^2}{y+1} \end{vmatrix}$$

$$=(y+1)\left(\frac{y^2}{y+1}\right) \begin{vmatrix} y+1 & \frac{y^3-3y-2}{y^2} \\ \frac{3y+2}{y+1} & \frac{y^2}{y+1} \end{vmatrix}$$

$$=(y+1)\left(\frac{y^2}{y+1}\right) \begin{vmatrix} y+1 & \frac{y^3-3y-2}{y^2} \\ 0 & \frac{y^2}{y+1}-\frac{(y^3-3y-2)(3y+2)}{y^2(y+1)^2} \end{vmatrix}$$

$$= (y+1)\left[\frac{y^2}{y+1}\right](y+1)\left[\frac{y^5+y^4-(y^3-3y-2)(3y+2)}{y^2(y+1)^2}\right]$$

$$= (y+1)\left[\frac{y^2}{y+1}\right](y+1)\left[\frac{y^5-2y^4-2y^3+9y^2+12y+4}{y^4+2y^3+y^2}\right]$$

$$= y^4 -3y^3 + y^2 + 8y +4.$$

V Theorem: The recursive procedure defined by the operations I through IV will reduce Sylvester's eliminant to the form:

$$R(y)\left.\begin{vmatrix} c & 0 & \cdots & & & 0 \\ 0 & c & 0 & \cdots & & 0 \\ 0 & 0 & c & 0 & \cdots & 0 \\ & & & \ddots & & \\ 0 & & \cdots & & & c \end{vmatrix}\right\}\text{ t rows},$$

Where R(y) is the product of rational functions of y, c is a rational function of y, and:

$$1 \leqslant t < n+s.$$

Proof: The procedure defined by the operations I through IV may be applied recursively to Sylvester's eliminant until either u-p = 0 or v-q = 0.

Let u-p = 0. Using the relations 1) and 2) in theorem IV, the remaining determinant contains b'_i ($i = q, q+1, \ldots , v$) in none of its rows and contains a'_i ($i = p = u$) in $v - q$ of its rows. Therefore Sylvester's eliminant is equal to:

$$R(y) \left.\begin{vmatrix} a'_u & 0 & 0 \ldots & 0 \\ 0 & a'_u & 0 \ldots & 0 \\ 0 & 0 & a'_u \ldots & 0 \\ & . & . & . \\ 0 & & \ldots & a'_u \end{vmatrix}\right\} \text{v-q rows}$$

$$= R(y)\ (a'_u)^{v-q}$$

Let $v-q = 0$. The remaining determinant contains a'_i ($i = p, p+1, \ldots, u$) in none of its rows and contains b'_j ($j = v = q$) in $u - p$ of its rows. Therefore Sylvester's eliminant is equal to:

$$R(y) \left.\begin{vmatrix} b'_v & 0 & 0 \ldots & 0 \\ 0 & b'_v & 0 \ldots & 0 \\ 0 & 0 & b'_v \ldots & 0 \\ & . & . & . \\ 0 & & \ldots & b'_x \end{vmatrix}\right\} \text{u-p rows}$$

$$= R(y)\ (b'_v)^{u-p}.$$

CHAPTER III

ALGORITHMS

In chapter II a recursive procedure for the reduction of Sylvester's eliminant is developed. The present chapter describes this procedure along with all the necessary supporting procedures in algorithmic form. The language used in this description is ALGOL 60.

From the examples in the preceding chapter the elements in the eliminant during the reduction are in general the quotient of two polynomials. The element $a_i (i = 0, 1, 2, \ldots, n)$ may be written in its full form as:

$$a_i = \frac{\alpha_{i1t}y^t + \alpha_{i1t-1}y^{t-1} + \ldots + \alpha_{i11}y^1 + \alpha_{i10}}{\alpha_{i2u}y^u + \alpha_{i2u-1}y^{u-1} + \ldots + \alpha_{i21}y^1 + \alpha_{i20}} .$$

The coefficients α_{iuj} are in general rational numbers and must be stored as ordered pairs of integers. In the following procedure the coefficients of the sequence of elements a_i $(i = 0, 1, 2, \ldots, n)$ are stored in the four dimensional array A[0:n, 1:2, 0:r, 1:2] where the first subscript with values 0 to n refers to the element of the sequence a_i $(i = 0, 1, 2, \ldots, n)$; the second subscript with values 1 and 2 refers respectively to numerator and denominator of the rational polynomial a_i; the third subscript with values 0 to r refers to the coefficient α_{i1j} of the polynomial; and the fourth subscript with values 1 and 2 refers to the elements of the ordered pair α_i. The coefficients of the elements b_j (j = 0, 1, 2, ... , s) are stored in the array B[0:s, 1:2, 0:r, 1:2]. The subscript bound r is less than: s (max degree a_i) + n (max degree b_j). As the reduction of the eliminant proceeds the storage arrays A and B are used to store the new elements of the reduced eliminant and r must be chosen to accommodate this storage.

Each of these polynomials has its own respective degree. The degree of the polynomials whose coefficients are stored in the array A are stored in the two dimensional array V[0:n, 1:2]. Thus the value V[i, 1] is the degree of the polynomial in the numerator of the quotient a_i above. The degree of the polynomial whose coefficients are stored in the array B are stored similarly in the array W[0:s, 1:2].

During the procedure various polynomials are stored in the four dimensional arrays C[1:1, 1:2, 0:r, 1:2], D[0:m, 1:2, 0:r, 1:2] and R[0:n+s, 1:2, 0:r, 1:2] where m is the greatest of n or s in the original set of equations. Degree storage for these arrays is in the integer arrays X[i:i, 1:2], Y[0:m, 1:2], Z[0:n=s, 1:2], respectively.

The procedure is written assuming that when the polynomials f(x, y) and g(x, y) are written as polynomials in x, the coefficients of each equation are of the form

$$\sum_{1=0}^{q} a_i y^i.$$

```
procedure Sylvester (A, B, n, s, r):
integer n, s, r;
integer array A, B;
     begin
     integer m;
     if n > s then m := n  else m := s;
          begin
          integer i, j, p, q, t, u, v;
          integer array C [1:1, 1:2, 0:r, 1:2];
                        D [0:m, 1:2, 0:r, 1:2];
                        R [0:n+s, 1:2, 0:r, 1:2];
                        V [0:n, 1:2];
                        W [0:s, 1:2];
                        X [1:1, 1:2];
                        Y [0:m, 1:2];
                        Z [0:n+s, 1:2];
          Boolean x;
comment  The arrays declared above correspond to the following
```

storage.

```
                                    V .. degree storage for A

                                    W .. degree storage for B

    C .. storage for polynomials,   X .. degree storage for C

    D .. storage for polynomials    Y .. degree storage for D

    R .. storage for polynomials    Z .. degree storage for R;

        procedure rat mult (I, i, L, J, j, M, K, k, n);

        comment I is a formal parameter corresponding to an
array for storage of polynomials. i corresponds to the subscript of
the element in the array I. L is a formal parameter corresponding to
an array for the storage of the degree of the polynomial referred to
by I and the subscript i. Similarly for the sets of three J. j, M and
K, k, n. I and J refer to the polynomials being multiplied and K to
the product;

    integer i, j, k ;

    array I, J, K, L, M, N ;

        begin

        mult (I, i, 1, L, J, j, 1, M, K, k, 1, N);

        mult (I, i, 2, L, J, j, 2, M, K, k, 2, N);

        com fact (K, k, 1, N, K, k, 2, N);

        N[k, 1]:= L[i, 1] + M[j, 1];

        N[K, 2]:= L[i, 2] + M[j, 2];

        end rat mult;
    procedure rat div (I, i, L, J, j, M, K, k, N);
    comment The correspondence between actual and formal
parameter in this procedure declaration is identical to that of the
procedure rat mult. The sets of three I, i, L; J, j, M; and K, k, N
refer to dividend, divisor, and quotient respectively;
    integer i, j, k;
    array I, J, K;
    integer array L, M, N;
        begin
        mult (I, i, 1, L, J, j, 2, M, K, k, 1, N);
        mult (I, i, 2, L, J, j, 1, M, K, k, 2, N);
        com fact (K, k, 1, N. K, k, 2, N);
        N[k, 1]:= L[i, 1] + M[j, 2];
        N[k, 2]:= L[i, 2] + M[j, 1];
        end rat div;
    procedure rat sub (I, i, L, J, j, M);
    comment This procedure subtracts the polynomial element with
subscript value corresponding to j which is stored in the array cor-
responding to the formal parameter J from the polynomial element
with subscript value corresponding to i which is stored in the array
corresponding to the formal parameter I and stores the result in the
array corresponding to the formal parameter I and giving it the
```

subscript value corresponding to i. The value for the degree stored in the array corresponding to the formal parameter L is corrected.

$$a_1/a_2 - b_1/b_2 = (a_1b_2 - b_1a_2)/a_2b_2$$

Let I and J correspond to storage for a_i/a_2 and b_1/b_2 respectively. In this procedure the polynomial product a_1b_2 is stored in the array I with subscript value corresponding to i. a_2b_1 is stored in the array C (declared at the beginning of the procedure Sylvester) since the storage corresponding to the array J must be left unchanged. Then the operation $a_1b_2 - a_2b_1$ is carried out;

```
integer i, j;
array I, J;
integer array L, M;
    begin
    mult (I, i, 1, L, J, j, 2, M, I, i, 1, L);
    mult (I, i, 2, L, J, j, 1, M, C, 1, 1, V);
    sub (I, i, 1, L, C, 1, 1, V);
    mult (I, i, 2, J, j, 2, M, I, i, 2, L);
    com fact (I, i, 1, L, I, i, 2, L);
    end rat sub;
```

comment The procedures mult, sub com fact, and degree operate on polynomials of the form:

$$\sum_{i=0}^{u} a[i]\ y \uparrow i$$

as opposed to the procedures above which are defined for the quotient of two such polynomials;

```
    procedure mult (I, i, u, L, J, j, v, M, K, k, w, N);
    comment I corresponds to an array for the storage of polynomials. i corresponds to the value of a subscript of a polynomial element stored in that array. u corresponds a real parameter which may have value 1 or 2 as to numerator or denominator of the polynomial referred to by the array and value corresponding to I and i. L corresponds to an array for degree storage for this polynomial. The correspondence in the parameter list for the sets of four J, j, v, M and K, k, w, N is the same. I, J, and K correspond to arrays which contain the two polynomials multiplied and the product respectively.
    integer i, j, k, u, v, w;
    array I, j, K;
    integer array L, M, N;
        begin
        integer array H[ 0:r, 1,2];
        integer m, t, a, e, f;
        a:= L[i, u];
        N[k, w] := a + M[j, v];
```

```
for m: = 0 step 1 until r do
    begin
    for t := 1,  2,  do
    H[m, t] := 0;
    end
for m := 0 step 1 until M[j, v]do
    begin
    for t := 0 step 1 until a do
        begin
        e := I[i, u, t, 1] × J[j, v, m, 1];
        f := I[i, u, t, 2] × J[j, v, m, 2];
        e := e × H[m+t, 2] + f × H[m+t, 1];
        f := f × H[m+t, 2];
        if e > f then C. F. (e, f) else C. F. (f, e);
        H[m+t, 1] := e
        H[m+t, 2] := f
        end
    end
for m := 0 step 1 until N[k, w]do
    begin
    for t := 1, 2 do
    K[k, w, m, t] := H[m, t];
```

```
            end
        end mult;
    procedure sub (I, i, u, L, J, j, v, M);
    comment I corresponds to an actual array for the storage of
polynomials, i to and integer parameter whose value is the subscript
of the polynomial, u corresponds to a parameter which may have
values 1 or 2 as to numerator or denominator of the polynomial ele-
ment referred to by the correspondents of I and i.  J, j, v, M corres-
pond to actual parameters in the same way.  The polynomial stored
in the array corresponding to J with subscript values corresponding
to j and v is subtracted from the polynomial stored in the array cor-
responding to I with subscript values i and u and the result is stored
in the array corresponding to I and is given the subscript values i and
u.  L and M correspond to degree storage and the value stored in the
array corresponding to L is corrected before exit from the procedure;
    integer i, j, u, v;
    integer array I, J, L, M;
        begin
        integer m, t, r, e, f;
        if L[i, u] >M[j, v] then m := L[i, u] else m := M[j, v];
        for t := L[i, u] + 1 step 1 until m do
        I[i, u, t, 1] := 0;
        for t := M[j, v] + 1 step 1 until m do
        J[j, v, t, 1] := 0;
        for t := 0 step 1 until m do
            begin
            e := I[i, u, t, 1] × J[j, v, t, 2] - I[i, u, t, 2] × J[j, v, t, 1];
            f := I[i, u, t, 2] × J[j, v, t, 2];
            if e > f then C. F. (e, f) else C. F. (f, e);
            I[i, u, t, 1] := e;
            I[i, u, t, 2] := f;
            if e ≠ 0 then r := t;
            end
        L[i, u] := r;
    procedure com fact (I, i, u, L, J, j, v, M);
```

comment Correspondence between the sets of formal parameters I, i, u, L and J, j, v, M is the same as described in the procedures mult and sub. This procedure finds any common factors of the polynomials stored in the arrays corresponding to i, u and j, v. Let a and b represent the polynomials stored in the arrays corresponding to I and J respectively. Let c be the common factor.

$$a = c\,a'$$

$$b = c\,b'$$

the common factor c is removed and a' and b' are stored in arrays corresponding to I and J in respective order.

```
    integer i, j, u, v;
    array I, J;
    integer L, M;
        begin
        integer m, r, s, t;
        array  E[0: L[i, u] + M[j, v], 1:2],
               F[0: L[i, u] + M[j, v], 1:2],
               G[0: L[i, u] + M[j, v], 1:2],
               H[0: L[i, u] + M[j, v], 1:2];
        m := L[i, u];
        s := M[j, v];
        if m > s then go to H1;
        for t := 0 step 1 until m do
            begin
            E[t, 1] := I[i, u, t, 1];
            E[t, 2] := I[i, u, t, 2];
            end
        for t := 0 step 1 until s do
            begin
            F[t, 1] := J[j, v, t, 1];
            F[t, 2] := J[j, v, t, 2];
        end
    go to H2;
H1: for t := 0 step 1 until m do
        begin
        F [t, 1] := I[i, u, t, 1] ;
        F [t, 2] := I[i, u, t, 2];
        end
    s := m;
    m := M[j, v];
    for t := 0 step 1 until m do
        begin
        E[t, 1] := J[j, v, t, 1];
        E[t, 2] := J[j, v, t, 2];
        end
```

comment This procedure is based on Euclid's algorithm for finding common factors. Let a and b represent the two given polynomials; also let r[i] and q[i] be polynomials. Assume the degree of the polynomial a is less than that of b.

$$\begin{array}{rcll} b & = & a\quad q[1] + r[1] & \qquad (1 \\ a & = & r[i]q[2] + r[2] & \qquad (2 \\ r[1] & = & r[1]q[3] + r[3] & \qquad (3 \\ \vdots & & \vdots & \\ r[n-1] & = & r[n]q[n+1] + r[n+1] & \qquad (n \\ r[n] & = & r[n+1]q[n+2] & \qquad (n+1 \end{array}$$

r[n+1] is the common divisor of a and b. The condition that the degree of the polynomial represented by a be less than that of the polynomial represented by b is satisfied by the if statements above. The following statements carry out the steps (1 through (n+1 stopping when the remainder r[n+2] = 0 is reached.

These steps may be accomplished recursively in the following manner. After step (1 denote the polynomial a by the name b and the polynomial r[1] by the name a. Divide the polynomial b by the polynomial a giving the remainder r[2]. After step i which yields the remainder r[i], denote r[i-1] by the name b and r[i] by the name a. Carry out the division above. Note that the actual parameters G and t are not used after the call in this case.

```
H2: div(F, s, E, m, G, t, H, r);
    if H[0, 1] = 0 ∧ v = 0 then go to H3;
    for t := 0 step 1 until m do
```

```
          begin
          F[t, 1] := E[t, 1];
          F[t, 2] := E[t, 2];
          end
     for t := 0 step 1 until r do
          begin
          E[t, 1] := H[t, 1];
          E[t, 2] := H[t, 2];
          end
     s::= m;
     m := r;
     go to H2;
     comment The following procedures divide out the common factors and correct the degree storage.  On entrance through the label
H3 the common factor is stored in the array E;
     for t := 0 step 1 until s do
  H3: s := L[i, u];
          begin
          F[t, 1] := I[i, u, t, 1];
          F[t, 2] := I[i, u, t, 2];
          end
     div (F, s, E, m, G, r, H, t);
     for t := 0 step i until r do
```

```
            begin
            I[i, u, t, 1] := G[t, 1];
            I[i, u, t, 2] := G[t, 2];
            end
        L[i, u] := r;
        s := M[j, v];
        for t := 0 step 1 until s do
            begin
            F [t, 1] := J[j, v, t, 1];
            F [t, 2] := J[j, v, t, 2];
            end
        div (F, w, E, m, G, r, H, t);
        for t := 0 step 1 until r do
            begin
            J [j, v, t, 1] := G[t, 1];
            J [j, v, t, 2] := G[t, 2];
            end
        M[j, v] := r;
        end com fact
        procedure div (I, a, J, b, P, c, R, d);
        comment The capital letters I, J, P, and R correspond to one
dimensional arrays for the storage of a single polynomial. They correspond
to arrays for the storage of dividend, divisor, quotient, and
remainder in the same order. The small letters correspond to values
for the degree of each polynomial. Thus the degree of the polynomial
stored in the array corresponding to I is the value of the parameter
corresponding to a;
    integer array I, J, P, R;
        begin
        integer i, e, f;
        integer array C[ 0:a, 1, 2];
    for t := 0 step 1 until a do
          begin
          C [t, 1] := I[t, 1];
          C [t, 2] := I[t, 2];
          end
    c:= a - b;
H1: e := C[a, 1] × J[b, 2];
    f := C[a, 2] × J[b, 1];
    if e >f then C. F. (e, f) else C. F. (f, e);
    P[a-b, 1] := e;
    P[a-b, 2] := f;
    for t := 0 step 1 until b do
          begin
          e := J[i, 1] × P[a-b, 1];
          f := J[i, 2] × P[a-b, 2];
          e := C[a-b+i, 1] × f - C [a-b+i, 2] × e;
          f := C[a-b+i, 2] × f;
          if e >f then C. F. (e, f) else C. F. (e, f);
          C[a-b+i, 1] := e;
          C[a-b+i, 2] := f;
          end
H2: a := a-1;
    if a < b then go to H3;
    if C[a, 1] ≠ 0 then go to H1;
    P[a-b, 1] := 0;
    P[a-b, 2] := 1;
```

```
          go to H2;
    H3: for i := 0 step 1 until b-1 do
          begin
          R[i, 1] := C[i, 1];
          R[i, 2] := C[i, 2];
          if C[i, 1] ≠ then d := i;
          end
       end div;
    procedure degree (I, i, u, L);
    comment I corresponds to a four-dimensional array for the
storage of polynomial elements, i is the subscript of the specific
polynomial element, u corresponds to an actual parameter which
has the value 1 or 2 corresponding to the numerator or denominator
of the polynomial element, and L corresponds to degree storage for
the array corresponding to I and r in this procedure is the r in the
parameter list of procedure Sylvester.  r is an upper bound on the
degree of the polynomials to be stored during the procedure Sylvester;
    integer i, u;
    array I;
    integer array L;
       begin
       integer t, v;
```

```
        for t:= 0 step 1 until r do
        if I[i,t,u,1] ≠ 0 then v := t;
        L[i,u] := v;
        end degree;
    procedure C.F. (a,b);
    comment  this procedure removes the common factors from
the integers a and b.  On entry to this procedure the integer corresponding
to b is less than the integer corresponding to a.  The procedure
is based on Euclid's algorithm. ;
    integer a,b;
        begin
        integer g,h,q,r;
        g := a;
        h := b;
     H1: r := [g - h × intier (g/h)]
        if r = 0 then go to H2;
        g := h;
        h := r;
        go to H1;
     H1: e := e/h;
        f := f/h;
        end C.F. ;
```

```
procedure string output (Q);
comment This procedure is an output for the string Q;
string Q;
   (code);


procedure alarm (x);
comment  This procedure is used to check after each step that
the storage placed in the arrays A and B has not exceeded the avail-
able space.  That is: the procedure checks to see that the degrees of
the various polynomials stored in these arrays has not exceeded the
value r;
Boolean x;
      begin
      integer i, j;
      for i := 1, 2 do
            begin
            for j := p step 1 until n do
            if V[j, i] > r then x := false;
            for j := q step 1 until s do
            if W[j, i] > r then x := false;
            end
      end alarm;
```

```
     comment All auxiliary procedures have been declared -- we
now start the compound tail for the main block of Sylvester;
     for i := 0 step 1 until n do
          begin
          degree (A, i, 1, V);
          V[i, 2] := 0;
          end
     for i := 0 step 1 until s do
          begin
          degree (B, i, 1, W);
          W[i, 2] := 0
          end
     p := q := 0;
     x := true ;
  L1: alarm (x);
     if x=false then string output (The degree of one of the polynomi-
     als stored in the arrays A and B has exceeded the value r.
     Choose a larger r);
     if n-p > s-q then go to L3;
     if V[n, 1] = 0∧A[n, 1, 0, 1] = 0 then go to L5;
     comment For purposes of description, let the storage in A be
represented by a[i] (i=p, p+1, . . . , n) and in B be represented by
```

by b[i] (i=q, q+1, . . . , s) as discussed in Chapter II. The author will assume this correspondence in the comments which follow. The two conditional statements above satisfy the conditions for case I. The statements up to but not including the one following the label L3 carry out the operations as described in case I, Chapter II;

```
    if W[s, 1] = 0 ∧ B[s, 1, 0, 1] = 0 then go to L2;
    rat div (B, s, W, A, n, V, C, 1, X);
    for i:= P step 1 until n-1 do
        begin
        rat mult (A, i, V, C, 1, W, D, i, Y);
        t:= s + n-1;
        rat sub (B, t, W, D, i, Y);
        end
    comment The following statements (not including L3 or there-
after) store the polynomial corresponding to a[n] in the array R and
store the degree in the array Z;
  L2:t := n-p+s-q;
    for i := 0 step 1 until V[n, 1] do
        begin
        R[t, 1, i, 1] := A[n, 1, i, 1];
        R[t, 1, i, 2] := A[n, 1, i, 2];
        end
    Z[t, 1] := V[n, 1];
    for i := 0 step 1 until V[n, 2] do
        begin
        R[t, 2, i, 1] := A[n, 2, i, 1];
        R[t, 2, i, 2] := A[n, 2, i, 2];
        end
    Z[t, 2] := V[n, 2];
    s := s-1 ;
    if s-q = 0 then go to L7;
    go to L1;
L3: if W[q, 1] = 0 ∧ B[q, 1, 0, 1] = 0 then go to L6;
    if V[p, 1] = 0 ∧ A[p, 1, 0, 1] = 0 then go to L4;
    comment The following statements (not including L5 or there-
after) carry out the operations as described in case III;
    rat div (A, p, V, B, q, W, C, 1, X);
    for i := q+1 step 1 until s do
        begin
        rat mult (B, i, W, C, 1, X, D, i, Y);
        t := i-q+p;
        rat sub (A, t, V, D, i, Y);
        end
    comment The following statements (not including L5 or there-
after) store the polynomial corresponding to b[q] inthe array R and
```

store the degree in the array Z;

```
  L4: t := n-p+s-q;
      for i := 0 step 1 until W[q, 1] do
          begin
          R[t, 1, i, 1] := B[q, 1, i, 1];
          R[t, 1, i, 2] := B[q, 1, i, 2];
          end
      Z[t, 1] := W[q, 1];
      for i := 0 step 1 until W[q, 2] do
          begin
          R[t, 2, i, 1] := B[q, 2, i, 1];
          R[t, 2, i, 2] := B[q, 2, i, 2];
          end
      Z[t, 2] := W[q, 2];
      p := p+1;
      if n-p = 0 then go to L8;
      go to L1;
  L5: if W[s, 1] = 0 ∧ B[s, 1, 0, 1} = 0 then go to L9;
```

comment The following statements (not including L6 or therethereafter) carry out the operations as described in case II. The expansion is made about the polynomial corresponding to the element b[s] and this polynomial is stored in the array R and the degree in

```
the array Z;
        t := n-p + s-q;
        for i := 0 step 1 until W[s, 1] do
            begin
            R[t, 1, i, 1] := B[s, 1, i, 1];
            R[t, 1, i, 2] := B[s, 1, i, 2];
            end
        Z[t, 1] := W[s, 1];
        for i := 0 step 1 until W[s, 2] do
            begin
            R[t, 2, i, 1] := B[s, 2, i, 1];
            R[t, 2, i, 2] := B[s, 2, i, s];
            end
        Z[t, 2] := W[s, 2];
        n := n-1;
        if n-p = 0 then go to L8;
        go to L1;
    L6: if V[p, 1] = 0 ∧ A[p, 1, 0, 1] = 0 then go to L9;
        comment The following statements (not including L7 or where
thereafter) carry out the operations as described in case IV. The expansion is made about the polynomial corresponding to the element
a[q] and this polynomial is stored in the array R and the degree is
```

```
stored in the array Z;

        t := n-p+s-q;

        for i := 0 step 1 until V[p, 1] do

            begin

            R[t, 1, i, 1] := A[t, 1, i, 1];

            R[t, 1, i, 2] := A[t, 1, i, 2];

            end

        Z[t, 1] := V[p, 1];

        for i := 0 step 1 until V[p, 2] do

            begin

            R[t, 2, i, 1] := A[t, 2, i, 1];

            R[t, 2, i, 2] := A[t, 2, i, 2];

            end

        Z[t, 2] := V[p, 2];

        q:= q:= 1;

        if s-q = 0 then go to L7;

        go to L1;
        comment  The following statements store the diagonal
matrix as shown in the proof of theorem V for the case s-q = 0;
    L7: t := n-p;
        u := W[s, 1];
        v := W[s, 2];
        for i := 0 step 1 until t do
            begin
            for j := 0 step 1 until u do
                begin
                R[i, 1, j, 1] := B[s, 1, j, 1];
                R[i, 1, j, 2] := B[s, 1, j, 2];
                end
            Z[i, 1] := u;
            for j := 0 step 1 until v do
                begin
                R[i, 2, j, 1] := B[s, 2, j, 1];
                R[i, 2, j, 2] := B[s, 2, j, 2];
                end
            Z[i, 2] := v
            end
        go to L10;
        comment  The following statements store the diagonal matrix
```

as shown in the proof of theorem V for the case n-p = 0;

```
   L8: t := s-q;
       u := V[n-1];
       v := V[n, 2];
       for i := 1 step 1 until t do
           begin
           for j := 0 step 1 until u do
               begin
               R[i, 1, j, 1] := A[n, 1, j, 1];
               R[i, 1, j, 2] := A]n, 1, j, 2];
               end
           Z[i, 1] := u;
           for j := 0 step 1 until v do
               begin
               R[i, 2, j, 1] := A[n, 2, j, 1];
               R[i, 2, j, 2] := A[n, 2, j, 2];
               end
           Z[i, 2] :=v
           end
       go to L10;
   L9: string output (Sylvester's eliminant vanishes identically. );
  L10: Alarm (x);
```

```
    if x =false then string output (The degree of one of the poly-
    nomials stored in the array R has exceeded the value r.
    Choose a larger r. );
    comment Sylvester's eliminant is the product of all the poly-
nomials stored in the array R.  The following statement reduces this
fraction to lowest terms.
    for i := n+s step (-1) until 1 do
        begin
        for j := n+s step 1 until 1 do
            begin
            if j = i go to H1;
            com fact (R, i, 1, Z, R, j, 2, Z);
        H1:
            end
        end
    end
end Sylvester;
```

BIBLIOGRAPHY

1. Dickson, Leonard Eugene. New first course in the theory of equations. London, John Wiley and Sons, Inc., 1949, 185 p.

2. Kurosh, A. G. Course of higher algebra. 5th ed., Moscow, State Publishing House for Technical-Theoretical Literature, 1956. Unpublished translation by Harry Goheen, Department of Mathematics, Oregon State University, Corvallis, Oregon. Chapter VII. (Hand-written)

3. Naur, Peter. Report on the Algorithmic language ALGOL 60. Association for Computing Machinery Communications 3(1960). pp.299-314.