AN ABSTRACT OF THE THESIS OF

JAMES JAY KEIZUR (Name) for the M. S. (Degree) in Mathematics (Major)

Date thesis is presented April 22, 1965

Title CONSTRUCTIVE METHODS FOR FINDING A ROOT OF CERTAIN ALGEBRAIC EQUATIONS WITH REAL COEFFICIENTS

Redacted for Privacy

Abstract approved ______________________
(Major professor)

A new procedure is developed for computing a root of algebraic equations with real coefficients and a degree n, where n is 2, 4, 6, 10, 14 or any positive odd integer.

A heuristic procedure is added to partially lift the restrictions on the degree n.

The procedures are written in the ALGOL 60 language.

CONSTRUCTIVE METHODS FOR FINDING A ROOT OF CERTAIN ALGEBRAIC EQUATIONS WITH REAL COEFFICIENTS

by

JAMES JAY KEIZUR

A THESIS

submitted to

OREGON STATE UNIVERSITY

in partial fulfillment of
the requirements for the
degree of

MASTER OF SCIENCE

June 1965

APPROVED:

Redacted for Privacy

Professor of Mathematics

In Charge of Major

Redacted for Privacy

Chairman of Mathematics Department

Redacted for Privacy

Dean of Graduate School

Date thesis is presented April 22, 1965

Typed by Carol Baker

ACKNOWLEDGEMENT

The author wishes to express his appreciation to his wife Julia for her support and to Dr. H. Goheen for his guidance in preparing this thesis.

TABLE OF CONTENTS

CONSTRUCTIVE METHODS FOR FINDING A ROOT OF CERTAIN ALGEBRAIC EQUATIONS WITH REAL COEFFICIENTS

CHAPTER I

INTRODUCTION

There have been many methods invented for computing the solution of algebraic equations. None of these methods however, are best suited for use with all algebraic equations. The engineer or the mathematician that is confronted with the task of computing an approximate root of a set of algebraic equations must take into account several factors before he can decide which method for solution of algebraic equations he should use. Some of the factors that he must consider are; the nature of the coefficients (real or complex), the degree of the equations that are to be solved, the accuracy of approximation that is required, etc.

A new method for solution of algebraic equations with real coefficients is developed in Chapters II and III. In Chapter II, a method is developed for computing an approximate root of an algebraic equation with real coefficients and a degree that is 2, 4, 6, 10, 14, or any positive odd integer. The restrictions on the degree may be partially lifted with the use of the method that was developed for Chapter III. This is a heuristic method that is designed to compute an approximate root of an algebraic equation with real coefficients and a degree that is any positive integer.

CHAPTER II

METHOD I

Method I may be used to compute at least one complex root of any algebraic equation that has, 1) real coefficients, and 2) a degree of n, where $n = 2, 4, 6, 10, 14,$ or any positive odd integer.

Case I: let

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

where $n = 2$ or any positive odd integer.

(i) If n is any positive odd integer, and the a_i's are real, then a real value α for x can be computed such that $f(\alpha) = 0$.

(ii) If $n = 2$, and the a_i's are complex, then two complex values β_1 and β_2 can be computed such that $f(\beta_1) = f(\beta_2) = 0$.

Standard procedures for computing the values α, β_1 and β_2 can be found in any standard numerical analysis text (2).

Case II: let

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

The a_i's are real; $a_n \neq 0$. $n = 4, 6, 10,$ or 14.

Although the case $n = 4$ has a classical history, it is included because it yields interesting results when the method that is developed for Chapter II is applied to it.

To begin the computation of roots of $f(x) = 0$, consider the following theorem.

<u>Theorem I:</u> if

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

where n is even, and the division algorithm is used to divide $f(x)$ by $(x^2 + q)$, thus giving

$$f(x) = Q(x, q)(x^2+q)+P(q)x+R(q),$$

then

$$P(q) = a_1 - a_3 q + a_5 q^2 - \ldots + (-1)^{\frac{n-2}{2}} a_{n-1} q^{\frac{n-2}{2}} , \qquad ,$$

and

$$R(q) = a_0 - a_2 q + a_4 q^2 - \ldots + (-1)^{\frac{n}{2}} a_n q^{\frac{n}{2}} .$$

Proof: if $n = 2$, then

$$f_2(x) = a_2(x^2+q)+a_1x+(a_0-a_2q) \quad .$$

$$f_2(x) = a_2x^2+a_2q+a_1x+a_0-a_2q \quad .$$

$$f_2(x) = a_2x^2+a_1x+a_0 \quad .$$

Suppose that k is any positive integer such that the following statement is true:

$$f_k(x) = Q_k(x,q)(x^2+q)+P_k(q)x+R_k(q) \; ,$$

where

$$f_k(x) = \sum_{i=0}^{k} a_ix^i \; ,$$

$$P_k(q) = a_1-a_3q+\cdots+(-1)^{\frac{k-2}{2}}a_{k-1}q^{\frac{k-2}{2}} \; ,$$

and

$$R_k(q) = a_0-a_2q+\cdots+(-1)^{\frac{k}{2}}a_kq^{\frac{k}{2}} \; . \tag{1}$$

Then, by letting $n = k+2$ in Theorem I, we obtain the following statement:

$$f_{k+2}(x) = Q_{k+2}(x,q)(x^2+q)+P_{k+2}(q)x+R_{k+2}(q) \; ,$$

where

$$f_{k+2}(x) = \sum_{i=0}^{k+2} a_ix^i \; ,$$

$$P_{k+2}(q) = P_k(q)+(-1)^{\frac{k}{2}}a_{k+1}q^{\frac{k}{2}} \quad ,$$

and

$$R_{k+2}(q) = R_k(q)+(-1)^{\frac{k+2}{2}} a_{k+2}q^{\frac{k+2}{2}} \quad . \tag{2}$$

Using statement (1), we will show that statement (2) is a true statement.

By definition,

$$f_{k+2}(x) = a_{k+2}x^{k+2}+a_{k+1}x^{k+1}+f_k(x) . \tag{3}$$

From (3), it follows that

$$f_{k+2}(x) = a_{k+2}x^{k+2}+a_{k+1}x^{k+1}+a_{k+2}qx^k+a_{k+1}qx^{k-1}$$

$$-a_{k+2}qx^k-a_{k+1}qx^{k-1}+f_k(x) . \tag{4}$$

By factoring (4) we get,

$$f_{k+2}(x) = (a_{k+2}x^k+a_{k+1}x^{k-1})(x^2+q)+f_k(x)$$

$$-q(a_{k+2}x^k+a_{k+1}x^{k-1}) . \tag{6}$$

The quantity $(a_{k+2}x^k+a_{k+1}x^{k-1})$ is a polynomial that has a degree of k and the coefficients a_i $(i = 0\cdots k)$ are all zero. Hence, we can use (1) to get,

$$(a_{k+2}x^{k}+a_{k+1}x^{k-1}) = Q(x,q)(x^{2}+q)+(-1)^{\frac{k-2}{2}}a_{k+1}q^{\frac{k-2}{2}}x$$

$$+ (-1)^{\frac{k}{2}}a_{k+2}q^{\frac{k}{2}} . \qquad (7)$$

Substituting the results of (1) and (7) into (6), we get

$$f_{k+2}(x) = (a_{k+2}x^{k}+a_{k+1}x^{k-1})(x^{2}+q)+Q_{k}(x,q)(x^{2}+q)$$

$$+P_{k}(q)x+R_{k}(q)+q[Q(x,q)(x^{2}+q)$$

$$-(-1)^{\frac{k-2}{2}}a_{k+1}q^{\frac{k-2}{2}}x-(-1)^{\frac{k}{2}}a_{k+2}q^{\frac{k}{2}}] . \qquad (8)$$

Simplifying (8), we get

$$f_{k+2}(x) = [(a_{k+2}x^{k}+a_{k+1}x^{k-1})+Q_{k}(x,q)+qQ(x,q)] \ (x^{2}+q)$$

$$+ (-1)^{\frac{k}{2}}a_{k+1}q^{\frac{k}{2}}x+P_{k}(q)x+(-1)^{\frac{k+2}{2}}a_{k+2}q^{\frac{k+2}{2}}+R_{k}(q). \qquad (9)$$

Let

$$Q_{k+2}(q,h) = [(a_{k+2}x^{k}+a_{k+1}x^{k-1})+Q_{k}(x,q)+qQ(x,q)] ,$$

then it follows from (9) that

$$f_{k+2}(x) = Q_{k+2}(q,h)(x^2+q)+((-1)^{\frac{k}{2}}a_{k+1}q^{\frac{k}{2}}+P_k(q))x$$

$$+ (-1)^{\frac{k+2}{2}} a_{k+2}q^{\frac{k+2}{2}} +R_k(q) .$$

Thus the induction principle assures us that Theorem I is true for all values for n where n is an even positive integer.

We use the division algorithm, dividing $f(x)$ by x^2+q to get:

$$f(x) = Q(x,q)(x^2+q)+P(q)x+R(q) .$$

From Theorem I, polynomials $P(q)$ and $R(q)$ can be expressed in terms of the a_i's and q. Computation of a value β for q such that $P(\beta) = R(\beta) = 0$ is the next task to complete. If $P(\beta) = R(\beta) = 0$, then $x = x \pm \sqrt{-\beta}$ are roots of $f(x) = 0$.

Before beginning the computation of a value β for q such that $P(\beta) = R(\beta) = 0$, the existence of such a β must be made certain. There are two possibilities for the existence of a value β for q such that $P(\beta) = R(\beta) = 0$, 1) $P(q)$ and $R(q)$ have a common factor $H(q)$, where the degree of $H(q) \geq 1$, or 2) either $P(q)$ or $R(q)$ vanishes identically. A test to determine if a value β for q exists, such that 1) or 2) is satisfied, can be constructed with the support of Theorem II.

Theorem II: the necessary and sufficient condition that $P(q)$ and $R(q)$ have a common factor $H(q)$, where the degree of $H(q) \geq 1$, is that $D_q(P,R) = 0$, $D_q(P,R)$ being Sylvester's eliminant for the polynomials $P(q)$ and $R(q)$.

Proof: the proof of Theorem II can be found in any standard theory of equations text (1).

The following example illustrates how Sylvester's eliminant is formed: let

$$f(x) = \sum_{i=0}^{6} a_i x^i .$$

The division algorithm is used, dividing $f(x)$ by x^2+q to get:

$$f(x) = Q(x,q)(x^2+q) + P(q)x + R(q).$$

It follows from Theorem I that

$$P(q) = a_5 q^2 - a_3 q + a_1 ,$$

and

$$R(q) = -a_6 q^3 + a_4 q^2 - a_2 q + a_0 .$$

$$D_q(P,R) = \begin{vmatrix} -a_6 & a_4 & -a_2 & a_0 & 0 \\ 0 & -a_6 & a_4 & -a_2 & a_0 \\ a_5 & -a_3 & a_1 & 0 & 0 \\ 0 & a_5 & -a_3 & a_1 & 0 \\ 0 & 0 & a_5 & -a_3 & a_1 \end{vmatrix}$$

It follows from the above example that if $P(q)$ or $R(q)$ vanishes identically, then $D_q(P,R) = 0$. It is noted that $R(q)$ cannot vanish identically if $a_6 \neq 0$. Now that a test is established for determining the existence of β, we may proceed to compute a value for β.

<u>Theorem III:</u> if $D_q(P,R) = 0$, and $n = 6$ or 4, then a value β for q can be computed such that $P(\beta) = R(\beta) = 0$.

Proof: from Theorem II, polynomials $P(q)$ and $R(q)$ have a common factor $H(q)$. This common factor can be computed by using the Euclidean Algorithm (4).

By repeated use of the division algorithm, the following sequence of equations is obtained:

$$
\begin{aligned}
R(q) &= Q(q)P(q) + r(q)\,, && \deg r(q) < \deg P(q),\\
P(q) &= Q_1(q)r(q) + r_1(q)\,, && \deg r_1(q) < \deg r(q)\,,\\
r(q) &= Q_2(q)r_1(q) + r_2(q)\,, && \deg r_2(q) < \deg r_1(q)\,,\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots && \cdots\cdots\cdots\cdots\\
r_{k-2}(q) &= Q_k(q)r_{k-1}(q)+r_k(q)\,, && \deg r_k(q) < \deg r_{k-1}(q)\,.
\end{aligned}
$$

The recurrence formula may be repeated until the absolute value of all of the coefficients of $r_{k-1}(q)$ are less than a small tolerance ϵ_1. Hence it follows that $r_{k-1}(q)$ is an approximate factor of $P(q)$ and $R(q)$. The polynomials $Q(q)$, $Q_1(q)$, $Q_2(q)$, $\cdots$ $Q_n(q)$, $r(q)$, $r_1(q)$, $r_2(q)$, $\cdots$ $r_k(q)$ can easily be computed by the usual process of long division of polynomials.

Once the common factor $H(q)$ of $P(q)$ and $R(q)$ has been approximated by $r_{k-1}(q)$, a value for β for q can be computed such that $r_{k-1}(\beta) = 0$. If $n = 4$ or 6, then by Theorem I, the degree of $R(q)$ is 2 or 3 respectively. Hence, the degree of $r_{k-1}(q) \leq 3$. Therefore Case I can be referenced to compute a complex value β such that $r_{k-1}(\beta) = 0$. If $r_{k-1}(\beta) = 0$, then $P(\beta) = R(\beta) = 0$.

<u>Corollary I:</u> if $D_q(P, R) = 0$, and $n = 4$ or 6, then

$$x = \pm\sqrt{-\beta}$$

are roots of $f(x) = 0$.

Proof: the proof of Corollary I follows immediately from Theorem III.

If however, $D_q(P, R) \neq 0$, then $f(x)$ can be expressed as follows:

$$f(x) = \sum_{i=0}^{n} c_i (x-h)^i ;$$

$$c_i = \frac{f^i(h)}{i!} .$$

We use the division algorithm, dividing $f(x)$ by x^2+q to get:

$$f(x) = Q_1(x, q, h)((x-h)^2+q)+P_1(q, h)(x-h)+R_1(q, h).$$

From Theorem I, polynomials $P_1(q, h)$ and $R_1(q, h)$ can be expressed in terms of the c_i's and q.

$$P_1(q, h) = c_1-c_3q+c_5q^2 - \cdots (-1)^{\frac{n-2}{2}} c_{n-1}q^{\frac{n-2}{2}} .$$

$$R_1(q, h) = c_0-c_2q+c_4q^2 - \cdots (-1)^{\frac{n-2}{2}} c_{n-2}q^{\frac{n-2}{2}} +(-1)^{\frac{n}{2}} c_n q^{\frac{n}{2}} .$$

The following determinant is an example of Sylvester's eliminant for the polynomials $P_1(q, h)$ and $R_1(q, h)$ when $n = 6$:

$$D_q(P_1, R_1) = \begin{vmatrix} -c_6 & c_4 & -c_2 & c_0 & 0 \\ 0 & -c_6 & c_4 & -c_2 & c_0 \\ c_5 & -c_3 & c_1 & 0 & 0 \\ 0 & c_5 & -c_3 & c_1 & 0 \\ 0 & 0 & c_5 & -c_3 & c_1 \end{vmatrix} .$$

To begin computation of an approximate complex root of $f(x) = 0$, consider Theorem IV.

Theorem IV: if $n = 4, 6, 10,$ or 14, then a complex value a for h can be computed such that $D_q(P_1, R_1) = 0$, and a complex value β for q can be computed such that $P_1(\beta, a) = R_1(\beta, a) = 0$.

Proof: from the fact that the c_i's are polynomials in h, it follows that $D_q(P_1, R_1)$ is a polynomial in h. The degree with respect to h of $D_q(P_1, R_1)$ is defined in terms of n by Theorem V.

Theorem V: if

$$P_1(q, h) = c_1 - c_3 q + c_5 q^2 - \cdots (-1)^{\frac{n-2}{2}} c_{n-1} q^{\frac{n-2}{2}} ,$$

and

$$R_1(q, h) = c_0 - c_2 q + c_4 q^2 - \cdots (-1)^{\frac{n-2}{2}} c_{n-2} q^{\frac{n-2}{2}} + (-1)^{\frac{n}{2}} c_n q^{\frac{n}{2}} ,$$

where

$$c_i = \frac{f^i(h)}{i!} , \quad f(x) = \sum_{i=0}^{n} a_i x^i ,$$

$c_n \neq 0$, $h \neq 0$, and n is a positive even integer, then the degree with respect to h of $D_q(P_1, R_1)$ is $\frac{n(n-1)}{2}$.

Proof: the proof of Theorem V depends on the following lemmas.

<u>Lemma 1:</u> the degree of $c_i(h)$ is $n-1$.

Proof: consider the term H_{ci} of $c_i(h)$ that contains the highest degree of h.

$$H_{ci} = \binom{n}{i} a_n h^{n-i} .$$

Since $a_n \neq 0$, it follows that the degree of c_i is $n-i$.

<u>Lemma 2:</u> if row (k) represents the k^{th} row of $D_q(P_1, R_1)$, then

$$\text{row (k)} = (T_1\lambda_{\ell-2}, T_2\lambda_{\ell-4}, \cdots, T_j\lambda_{\ell-2j}, \cdots, T_{n-1}\lambda_{\ell-2n+2}),$$

λ and ℓ being defined as below.

Proof:

(i) If k is a positive integer $0 < k \leq \frac{n}{2} - 1$, then $\ell = n+2k$, $T_j = 0$ when $\frac{n}{2} + k < j < k$, $T_j = (-1)^{\frac{n}{2}+(j-1)}$ when $\frac{n}{2} + k < j < k$, and $\lambda_{\ell-2j} = c_{\ell-2j}(h)$.

$$\text{row }(k) = (\overbrace{0,\cdots 0}^{k-1}, \pm c_n, \mp c_{n-2}, \cdots -c_2, c_0, \overbrace{0\cdots 0}^{\frac{n}{2}-k-1}).$$

(ii) $\frac{n}{2} < k < n-1$, then $\ell = 2k+1$, $T_j = 0$ when $k < j < k - \frac{n}{2}$, $T_j = (-1)^{\frac{n}{2}+j}$ when $k \geq j \geq k - \frac{n}{2}$, and $\lambda_{\ell-2j} = c_{\ell-2j}(h)$.

$$\text{row }(k) = (\overbrace{0,\cdots 0}^{k-\frac{n}{2}}, \mp c_{n-1}, \pm c_{n-3}, \cdots, -c_3, c_1, \overbrace{0\cdots 0}^{n-k-1}).$$

<u>Lemma 3:</u> If $h \neq 0$, then $D_q(P_1, R_1)$ can be expressed as a determinant such that the elements in a column all have the same degree with respect to h.

Proof: from Lemma 2,

$$\text{row }(k) = (T_1\lambda_{\ell-2}, \cdots, T_j\lambda_{\ell-2j}, \cdots, T_{n-1}\lambda_{\ell-2n+2}) .$$

Let s be the degree with respect to h of $\lambda_{\ell-2j}$. From Lemma 1,

$$s = n - \ell + 2j .$$

Let

$$\text{row }(k) = \frac{1}{h^\ell}(h^\ell T_1\lambda_{\ell-2}, \cdots, h^\ell T_j\lambda_{\ell-2j}, \cdots, h^\ell T_{n-1}\lambda_{\ell-2n+2}) .$$

The degree with respect to h of $h^\ell T_{n-1}\lambda_{\ell-2n+2}$ is $n+2j$. The degree of $h^\ell T_j\lambda_{\ell-2j}$ with respect to h is not dependent on k. Therefore Lemma 3 is proved.

$D_q(P_1, R_1)$ is a determinant and therefore, the rules for evaluating a determinant apply. $D_q(P_1, R_1)$ may be reduced to the sum of products of the elements of $D_q(P_1, R_1)$. With this idea, consider Lemma 4.

Lemma 4: the degree with respect to h of $D_q(P_1, R_1)$ cannot be greater than $\frac{n}{2}(n-1)$.

Proof: from Lemma 3 and the rules for evaluating a determinant, the terms of the reduced determinant $D_q(P_1, R_1)$ all have the same degree with respect to h. Therefore, the degree with respect to h of $D_q(P_1, R_1)$ cannot be greater than the degree of any one term of the reduced determinant $D_q(P_1, R_1)$. A convenient term of the reduced determinant of $D_q(P_1, R_1)$ is the product of the elements on the diagonal of the determinant of $D_q(P_1, R_1)$. This term may be expressed as follows:

$$\prod_{k=1}^{\frac{n}{2}-1} \left(\frac{h^{n+2k}}{h^{n+2k}} \lambda_n \right) \prod_{k=\frac{n}{2}}^{n-1} \left(\frac{h^{2k+1}}{h^{2k+1}} \lambda_1 \right) \tag{11}$$

From Lemma 1, the degree with respect to h of λ_n and λ_1 is 0 and $n-1$ respectively. Thus $\frac{n}{2}(n-1)$ is the maximum degree with respect to h of $D_q(P_1, R_1)$.

Lemma 5: the degree with respect to h of $D_q(P_1, R_1)$ cannot be less than $\frac{n(n-1)}{2}$.

Proof: in the proof of Lemma 3 and 4, it has been established that $D_q(P_1, R_1)$ may be reduced to the sum of terms that all have the same degree with respect to h. From Lemma 4, the maximum degree with respect to h of these terms is $\frac{n(n-1)}{2}$. Consider the term $H \cdot h^{\frac{n(n-1)}{2}}$ which is the term of highest degree of $D_q(P_1, R_1)$ unless $H = 0$. To further define H, we introduce the following equation:

$$J(h) = a_n (1+h)^n .$$

The division algorithm may be used to get:

$$J(h) = Q_j(q, h)(h^2+q) + P_j(q)h + R_j(q) .$$

From Theorem I,

$$P_j(q) = a_n(\binom{n}{1} - \binom{n}{3}q + \binom{n}{5}q^2 \cdots (-1)^{\frac{n-2}{2}} \binom{n}{n-1} q^{\frac{n-2}{2}}) ,$$

and

$$R_j(q) = a_n(\binom{n}{0} - \binom{n}{2}q + \binom{n}{4}q \cdots (-1)^{\frac{n-2}{2}} \binom{n}{n-2} q + (-1)^{\frac{n}{2}} \binom{n}{n} q^{\frac{n}{2}}) . \quad (12)$$

From Lemma 1,

$$H_{ci} = \binom{n}{i} a_n h^{n-i} \quad (13)$$

It follows from (12), (13), and the definition of Sylvester's eliminant for the polynomials $P_j(q)$ and $R_j(q)$, that the term

$$D_q(P_j, R_j) \cdot a_n^{n-1} \cdot h^{\frac{n(n-1)}{2}} \equiv H \cdot h^{\frac{n}{2}(n-1)} .$$

For example: let $n = 4$, then

$$J(h) = a_4(1+h)^4 .$$

From (12),

$$P_j(h) = a_4(\binom{4}{1} - \binom{4}{3}q),$$

and

$$R_j(h) = a_4(\binom{4}{0} - \binom{4}{2}q \binom{4}{4}q).$$

Hence,

$$D_q(P_j, R_j) = \begin{vmatrix} \binom{4}{4} a_4 & -\binom{4}{2} a_4 & \binom{4}{0} a_4 \\ -\binom{4}{3} a_4 & \binom{4}{1} a_4 & 0 \\ 0 & -\binom{4}{3} a_4 & \binom{4}{1} a_4 \end{vmatrix} .$$

The term of highest degree with respect to h of $D_q(P_1, R_1)$ is:

$$H = \begin{vmatrix} H_{c4} & -H_{c2} & H_{c0} \\ -H_{c3} & H_{c1} & 0 \\ 0 & -H_{c3} & H_{c1} \end{vmatrix} .$$

From (13),

$$H = \begin{vmatrix} \binom{4}{4}a & -\binom{4}{2}ah & \binom{4}{0}ah \\ -\binom{4}{3}ah & \binom{4}{1}ah & 0 \\ 0 & -\binom{4}{3}ah & \binom{4}{1}ah \end{vmatrix} .$$

By reducing H, we get:

$$H = a_4^3 \cdot \binom{4}{4} \cdot \binom{4}{1}^2 . h^6 + a_4^3 \cdot \binom{4}{3}^2 \cdot \binom{4}{0} h^6 - a_4^3 \cdot \binom{4}{2} \cdot \binom{4}{3} \cdot \binom{4}{1} h .$$

Hence,

$$H = a_4^3 \cdot D_q(P_j, R_j)h .$$

If $D_q(P_j, R_j) = 0$, then from Corollary I, h^2+q is a factor of $a_n(1+h)^n$. Hence,

$$h^2+q \equiv h^2+2h+1 .$$

This is a contradiction. Thus $D_q(P_j, R_j) \neq 0$, and by definition $a_n \neq 0$. Therefore, the degree with respect to h of $D_q(P_1, R_1)$ cannot be less than $\frac{n(n-1)}{2}$.

Lemma 4 states that the degree with respect to h of $D_q(P_1, R_1)$ cannot be greater than $\frac{n(n-1)}{2}$ and Lemma 5 states that the degree with respect to h of $D_q(P_1, R_1)$ cannot be less than $\frac{n(n-1)}{2}$. Therefore, the degree with respect to h of

$D_q(P_1, R_1)$ must be exactly $\frac{n(n-1)}{2}$.

We will continue with the proof of Theorem IV by considering each value for n separately.

(i) If $n = 6$, then from Theorem V, the degree with respect to h of $D_q(P_1, R_1)$ is 15. Therefore, Case I can be referenced to compute a real value α for h such that $D_q(P_1, R_1) = 0$. If $D_q(P_1, R_1) = 0$, then Theorem III can be referenced with $P_1(q, \alpha)$ as $P(q)$ and $R_1(q, \alpha)$ as $R(q)$ to compute a complex value β for q such that $P_1(\beta, \alpha) = R_1(\beta, \alpha) = 0$. Hence $x = \alpha \pm \sqrt{-\beta}$ are roots of $f(x) = 0$.

(ii) If $n = 4$, then from Theorem V, the degree with respect to h of $D_q(P_1, R_1)$ is 6. Therefore, (i) of Theorem IV can be used to compute a complex value α for h such that $D_q(P_1, R_1) = 0$. Let

$$\alpha = \alpha_1 \pm \sqrt{-\beta}$$

From Theorem I, the degree with respect to q of $R(q, \alpha)$ is 2. Hence, Case I can be referenced to compute two values ϕ_1 and ϕ_2 for q such that $R_1(\phi_1, \alpha) = R_1(\phi_2, \alpha) = 0$. If $D_q(P_1, R_1) = 0$, then either

$P_1(\phi_1, \alpha) = R_1(\phi_1, \alpha) = 0$ or $P_1(\phi_2, \alpha) = R_1(\phi_2, \alpha) = 0$. Hence a complex value β for q can be computed such that $P_1(\beta, \alpha) = R_1(\beta, \alpha) = 0$, thus making $x = \alpha \pm \sqrt{-\beta}$ roots of $f(x) = 0$.

(iii) If $n = 10$ or 14, then from Theorem V the degree with respect to h of $D_q(P_1, R_1)$ is 45 or 91 respectively. Therefore Case I can be referenced to compute a real value α such that $D_q(P_1, R_1) = 0$. Hence, from Theorem II, $P_q(q, \alpha)$ and $R(q, \alpha)$ have a common factor $H(q, \alpha)$. An approximation of this common factor can be computed by using the Euclidean algorithm as described in the proof of Theorem III. If $n = 10$ or 14, from Theorem I the degree with respect to q of $R_q(q, \alpha)$ is 5 or 7. Hence the degree with respect to q of $H(q, \alpha) < 7$. If the degree with respect to q of $H(q, \alpha)$ is $1, 2, 3, 5$, or 7 then Case I can be referenced to compute a complex value β for q such that $R_1(\beta, \alpha) = P_1(\beta, \alpha) = 0$. If the degree of $H(q, \alpha)$ is 4 or 6, then (i) or (ii) of Theorem IV may be referenced to compute a complex value β for q such that $H(\beta, \alpha) = 0$. This yields $x = \alpha \pm \sqrt{-\beta}$ as roots of $f(x) = 0$.

CHAPTER III

METHOD II

Method II is offered as a heuristic method for computing a complex root of an algebraic equation with real coefficients.

Case I of Method I may be referenced to compute a real root of any algebraic equation that has an odd degree and real coefficients. Hence, only algebraic equations that have an even degree and real coefficients will be considered. Let

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

where the a_i's are real, and n is a positive even integer. As in Method I, the division algorithm is used, dividing $f(x)$ by x^2+q to get:

$$f(x) = Q(x,q)(x^2+q)+P(q)x+R(q) .$$

From Theorem I, the polynomials $P(q)$ and $R(q)$ may be expressed in terms of the a_i's and q.

$$P(q) = a_1-a_3q+a_5q^2 - \cdots (-1)^{\frac{n-2}{2}} a_{n-1}q^{\frac{n-2}{2}} .$$

$$R(q) = a_0-a_2q+a_4q^2 - \cdots (-1)^{\frac{n-2}{2}} a_{n-2}q^{\frac{n-2}{2}} +(-1)^{\frac{n}{2}} a_nq^{\frac{n}{2}} .$$

If the Sylvester's eliminant $D_q(P,R)$ for the polynomials $P(q)$ and $R(q)$ is zero, then from Theorem II, the polynomials $P(q)$ and $R(q)$ have a common factor $H(q)$. The assumption of the heuristic is that $H(q)$ is linear with respect to q. The Euclidean Algorithm as described in the proof of Theorem III may be used to compute an approximation $r_{k-1}(q)$ of $H(q)$. $r_{k-1}(q)$ is linear; hence, a real value α can be computed such that $r_{k-1}(\alpha) = 0$, thus making $x = \pm\sqrt{-\alpha}$ approximate roots of $f(x) = 0$.

If $D_q(P,R) \neq 0$, then $f(x)$ may be expressed as follows:

$$f(x) = \sum_{i=0}^{n} c_i(x-h)^i ,$$

where

$$c_i = \frac{f^i(h)}{i!} .$$

The division algorithm may be used, dividing $f(x)$ by $(x-h)^2+q$ to get,

$$f(x) = Q_1(x,q,h)((x-h)^2+q)+P_1(q,h)(x-h)+R_1(q,h).$$

From Theorem I, the polynomials $P_1(q,h)$ and $R_1(q,h)$ may be expressed in terms of the c_i's and q.

$$P_1(q,h) = c_1 - c_3q + c_5q^2 - \cdots (-1)^{\frac{n-2}{2}} c_{n-1} q^{\frac{n-2}{2}} .$$

$$R_1(q,h) = c_0 - c_2q + c_4q^2 - \cdots (-1)^{\frac{n-2}{2}} c_{n-2} q^{\frac{n-2}{2}} + (-1)^{\frac{n}{2}} c_n q^{\frac{n}{2}} .$$

To compute an approximate root of $f(x) = 0$, we must compute a value α for h and a value β for q such that $P_1(\beta, \alpha) = R_1(\beta, \alpha) = 0$. Consider the following theorem:

Theorem VI: the necessary and sufficient condition that $P_1(q,\lambda)$ and $R_1(q,\lambda)$ have a common factor $H(q)$, where the degree of $H(q) \geq 1$, and λ is a complex number, is that $D_q(P_1, R_1) = 0$. $D_q(P_1, R_1)$ is the Sylvester's eliminant for the complex polynomials $P_1(q,\lambda)$ and $R_1(q,\lambda)$.

Proof: the proof of Theorem VI is equivalent to the proof of Theorem II.

Lemma 6: if a complex value λ for h is computed such that $D_q(P_1, R_1) = 0$, then a complex value β for q can be computed such that $P_1(\beta, \lambda) = R_1(\beta, \lambda) = 0$.

Proof: if $D_q(P_1, R_1) = 0$, then from Theorem VI, the polynomials $P_1(q,\lambda)$ and $R_1(q,\lambda)$ have a common factor $H(q)$. The Euclidean algorithm as described in the proof of Theorem III may be used to compute an approximation $r_{k-1}(q)$ of $H(q)$.

According to the assumption of the heuristic, $r_{k-1}(q)$ is linear; hence, a complex value β for q can be computed such that $r_{k-1}(\beta) = 0$. Therefore, $H(\beta) = 0$; thus making $P_1(\beta,\lambda) = R_1(\beta,\lambda) = 0$.

The Sylvester's eliminant $D_q(P_1, R_1)$ is a polynomial in h with real coefficients. If the degree of $D_q(P_1, R_1)$ with respect to h is odd, then a real value α for h can be computed such that $D_q(P_1, R_1) = 0$. Hence, we will assume that the degree of $D_q(P_1, R_1)$ with respect to h is even. Consider treating the polynomial $D_q(P_1, R_1)$ as a new problem and applying the same sequence of steps to it as was applied to $f(x)$. Using the Euclidean algorithm, and dividing $D_q(P_1, R_1)$ by $((h-h_1)^2+q_1)$ we get,

$$D_q(P_1, R_1) = Q_2(h, h_1, q_1)((h-h_1)^2+q_1)+P_2(q_1, h_1)(h-h_1)+R_2(q_1, h_1) .$$

It follows from Lemma 6, that if a complex root of $D_q(P_1, R_1) = 0$ can be computed, then a complex root of $f(x) = 0$ can be computed. Thus the following sequence of equations is suggested:

$$\begin{aligned}
f(x) &= Q_1(x, h, q)((x-h)^2+q)+P_1(q, h)(x-h) + R_1(q, h) , \\
D_q(P_1, R_1) &= Q_2(h, h_1, q)((h-h_1)^2+q_1)+P_2(q_1, h_1)(h-h_1)+R_2(q_1, h_1) , \\
D_q(P_2, R_2) &= Q_3(h_1, h_2, q_2)((h_1-h_2)^2+q_2)+P_3(q_2, h_2)(h_1-h_2)+R_3(q_2, h_2), \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
D_q(P_{k-1}, R_{k-1}) &= Q_k(h_{k-2}, h_{k-1}, q_{k-1})((h_{k-2}-h_{k-1})^2+q_{k-1}) \\
&\quad +P_k(q_{k-1}, h_{k-1})(h_{k-2}-h_{k-1})+R_k(q_{k-1}, h_{k-1}) . \qquad (14)
\end{aligned}$$

From Theorem V, and the sequence of equations (14), the degree with respect to h_{k-1} of $D_q(P_k, R_k)$ is n_k, where

$$n_k = \frac{n_{k-1}(n_{k-1}-1)}{2} \quad .$$

<u>Theorem VII:</u> if

$$n_p = \frac{n_{p-1}(n_{p-1}-1)}{2} \quad ,$$

then

$$n_p = 2^{k-p}\ell_p \quad ,$$

where $p \leq k$, p and k are positive integers, and ℓ_p is a positive odd integer.

Proof: let

$$n_0 = 2^k \ell_0 \, , \tag{15}$$

where ℓ_0 is a positive odd integer. If $p = 1$, then

$$n_1 = \frac{n_0(n_0-1)}{2} \quad . \tag{16}$$

From (15) and (16)

$$n_1 = 2^{k-1}\ell_0(2^k\ell_0-1) \, ,$$

thus

$$n_1 = 2^{k-1}\ell_1 \quad ,$$

where ℓ_1 is a positive odd integer. If $P = k-1$, and

$$n_{k-1} = 2\ell_{k-1} , \tag{17}$$

where ℓ_{k-1} is a positive odd integer, then we shall prove that

$$n_k = \ell_k ,$$

where ℓ_k is a positive odd integer. By definition,

$$n_k = \frac{n_{k-1}(n_{k-1}-1)}{2} . \tag{18}$$

From (17) and (18),

$$n_k = \frac{2\ell_{k-1}(2\ell_{k-1}-1)}{2} . \tag{19}$$

By simplifying (19) we get,

$$n_k = \ell_{k-1}(2\ell_{k-1}-1)$$

$$n_k = \ell_k .$$

From Theorem VII, the degree with respect to h of $D_q(P_k,R_k)$ is odd. Therefore, from Case I of Chapter II, a real value α for h in the sequence (14) can be computed such that $D_q(P_k,R_k) = 0$. Consider the following theorem:

<u>Theorem VIII:</u> if in the sequence (14) $D_q(P_s, R_s) = 0$, where

$1 > s \geq k$, then a complex value β_{s-1} for q_{s-1} can be computed such that $D_q(P_{s-1}, R_{s-1}) = 0$.

Proof: let λ be a complex value for h_{s-1} such that $D_q(P_s, R_s) = 0$. From Theorem VI, the polynomials $P_s(q_{s-1}, \lambda)$ and $R_s(q_{s-1}, \lambda)$ have a common factor $H(q)$, which may be computed by the Euclidean algorithm. According to the assumption of the heuristic this polynomial is assumed to be linear; hence, a value β_{s-1} for q_{s-1} can be computed such that $H(\beta_{s-1}) = 0$. If $H(\beta_{s-1}) = 0$, then $P_s(\beta_{s-1}, \lambda) = R_s(\beta_{s-1}, \lambda) = 0$. If $P_s(\beta_{s-1}, \lambda) = R_s(\beta_{s-1}, \lambda) = 0$, then it follows from the sequence (14) that the values $\lambda \pm \sqrt{-\beta_{s-1}}$ for h_{s-2} are complex roots of $D_q(P_{s-1}, R_{s-1}) = 0$.

From Theorem VIII, it follows that the complex values

$$x = \alpha \pm \sqrt{-\beta_{k-1}} \pm \sqrt{-\beta_{k-2}} \pm \cdots \pm \sqrt{-\beta_1} \pm \sqrt{-\beta}$$

are complex roots of $f(x) = 0$, where α is a real root of $D_q(P_k, R_k) = 0$, $\alpha \pm \sqrt{-\beta_{k-1}}$ are complex roots of $D_q(P_{k-1}, R_{k-1}) = 0$, etc. through sequence (14).

BIBLIOGRAPHY

1. Dickson, Leonard Eugene. New first course in the theory of equations. London, Wiley, 1949. 185 pp.

2. Herriot, John G. Methods of mathematical analysis and computation. London, Wiley, 1963. 198 pp.

3. Holroyd, John Ries. Algorithms for the solution of two algebraic equations in two unknowns. Master's thesis. Corvallis, Oregon State University, 1962. 47 numb. leaves.

4. McCoy, Neal H. Introduction to modern algebra. Boston, Allyn and Bacon, 1961. 304 pp.

5. Naur, Peter. Report on the algorithmic language ALGOL 60. Communications of the Association for Computing Machinery 3:299-314. 1960.

APPENDICIES

APPENDIX A

In Chapter II and III a method for computing an approximate root of an algebraic equation with real coefficients was developed. To describe this method in a form that is easily adapted to a computer program, the procedures are written in the ALGOL 60 language.

The procedures, Sylvester, ratmult, mult, comfact, and C. F., may be found in Appendix B. They are copied, with the exception of a few minor adjustments, from a thesis written by John Holroyd (3).

<u>procedure</u> comroot(a, N, r) results: (b);

<u>integer</u> <u>array</u> a;

<u>integer</u> N, r;

<u>real</u> <u>array</u> b;

<u>begin</u> <u>comment</u>: the procedure comroot will compute an approximate

complex root of the algebraic equation

$$f(y) = 0,$$

where

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

the values for the formal parameter a are stored in the

integer array a[0:N, 1:2]. The values $i(i = 0, 1, \cdots, N)$

for the first subscript of the array a refer to the coefficient a_i of y^i. The coefficients $a_i (i = 0, 1, \cdots, N)$ are rational numbers and are stored as ordered pairs of integers. The values 1 and 2 for the second subscript refer to the numerator and denominator respectively of the coefficient specified by the value of the first subscript. N contains the degree of the polynomial $f(y)$. The formal parameter b stands for an approximate complex root of $f(y) = 0$. The values for b are stored in the integer array b[1:2]. The values 1 and 2 for the subscript of the array b refer to the real and imaginary part respectively of the complex root. The integer r is n_{k-1} in Theorem VII of Chapter III. ;

```
integer NS, i, k, m, s, t, ss1, ss2, ss3, ss4;
integer array I11[1:2], I12[1:2], I13[1:2], rs[1:2], ss[1:4],
I21[0:r, 1:2], I22[0:r, 1:2], as[0:r×(r-1)/2, 1:2],
c[0:r×(r-1)/2, 0:r×(r-1)/2, 1:2], P[0:(r-2)/2, 1:2, 0:r-1, 1:2],
R[0:r/2, 1:2, 0:r, 1:2], PS[1:N, 0:(r-2)/2, 0:r-1, 1:2],
RS[1:N, 0:r/2, 0:r, 1:2], S[0:r, 1:2, r×(r-1)/2, 1:2];
array T11[1:2], T12[1:2], T21[0:r, 1:2], T22[0:r, 1:2];
real eps;
switch return, stop, b1, b2, b3;
for i:=0 step 1 until N do
for k:=1 step 1 until 2 do
```

```
    as[i, k] := a[i, k] ;
    NS:= N ;
    for i:= 1 step 1 until 4 do
    ss[i] := 1 ;
    eps: .0000001 ;
St: if (NS-entier(NS/2)) ≠ 0 then
    begin comment: the degree of f(y) is odd. Hence we can
            refer to Case I(i) of Chapter II to compute a real root
            of f(y) = 0. ;
            realroot (eps, f, as, NS) results: (I11) ;
            b[1] := I11[1]/I11[2] ;
            b[2] := 0 ;
            ss1:= ss[1] ;
            go to return (ss[1] ) ;
    end ;
    if NS = 2 then
    begin comment: the degree of f(y) is 2. Hence we can
            refer to Case I(ii) of Chapter II to compute a complex
            root of f(y) = 0. ;
            for i:= 0 step 1 until 2 do
            begin T21[i, 1] := as[i, 1]/as[i, 2] ;
                  T21[i, 1] := 0 ;
            end i ;
            case ii (T21) results: (b, T12) ;
```

```
        go to return (ss[1] );
end ;
if  NS ≥ 16 then go to  M2;
comment:  form D_q(P, R) ;
for i:=0  step 1 until (NS-2)/2 do
begin for k:=1  step 1 until 2 do
        P[0, 1, i, k] := as[2×i-1, k] ×(-1)↑ i;
        C. F. (P[0, 1, i, 1] , P[0, 1, i, 2] );
end k;
for i:=0  step 1 until NS/2 do
begin for k:= 1  step 1 until 2 do
        R[0, 1, i, k] := as[ 2×i,  k]×(-1)↑ i;
        C. F. (R[0, 1, i, 1] , R[0, 1, i, 2] ;
end k;
i:= 0;
k:= 0;
m:= 1;
Sylvester (R, P, i, k, m) results: (S, I21);
if (S[0, 1, 0, 1] x S[0, 2, 0, 2] ) ≠ 0  then go to  C1;
comment:  D_q(P,R) = 0,  hence we will compute the common
factor of  P(q)  and  R(q)  and then compute an approximate
root of the common factor, ;
I21[0, 1] := (NS-2)/2;
```

```
    I22[0, 1] := NS/2;
    comfact (R, 0, 1, I22, P, 0, 1, I21) results: (as, NS);
    comment: the common factor of  P  and  R  is now in the
    array  as  and the degree of the common factor is in  NS.
    We will set the return switches to  b1  and go back to the
    start. ;
    ss[1] := 2;
    ss[3] := 2;
    ss[4] := 2;
    ss2:= ss[2] ;
    go to  St;
bl: comment:  b  contains an approximate complex root of the
    common factor of  P(q)  and  R(q).  Therefore

                          y = √-b

    is an approximate root of  f(y) = 0. ;
    for i:= 1  step 1 until 2 do
    b[i] ;= -b[i] ;
    csqrt (b) results: (b) ;
    go to return (ss2) ;
C1: if NS = 8  then go to M2 else
    if NS = 12  then go to M2;
    comment: Sylvester's eliminant for  P(q)  and  R(q)  is
```

not zero. Therefore we will proceed with Theorem IV of Chapter II;

```
cees (as, NS) results: (c) ;
form PR(c, NS) results: (P,R);
m:= NS/2;
s := (NS-2)/2;
t := NSx(NS-1)/2;
Sylvester (R,P,m,s,t) results: (S, I21);
for i:= 0 step 1 until NS-2 do
m:= i+1;
ratmult (S,i,I21,S,m,I21,S,m,I21);
m:= NS-1;
comfact (S,m,1,I21,S,m,2,I21) results: (I22, k)
if NS = 4 then go to C2;
```

comment: the degree of Sylvester's eliminant for $P_1(q,\alpha)$ and $R_1(q,\alpha)$ is odd. Hence we will compute an approximate real root of $D_q(P_1, R_1) = 0.$;

```
for i:= 0 step 1 until I21[m,1] do
for k:= 1 step 1 until 2 do
I22[i,k] := S[m,1,i,k] ;
k:= I21[m,1] ;
realroot (eps,f,I22,k) results: (rs);
```

comment: rs contains an approximate real root of

```
Dq(P1,R1) = 0. ;

comment: we can now evaluate ci(rs) (i=0, 1, ···, N). ;

for i:=0  step 1 until NS do

begin for k:= 0  step  1  until  Ns-i  do

        for m:= 1  step  1  until  2  do

        I21[k, m] := c[i, k, m] ;

        m:= N-i;

        f(I21, m, rs) results:  (I11);

        I21[i, 1] := I11[i] ;

        I21[i, 2] ;  I11[2] ;

end  i;

comment: form  P1(q, rs)   and   R1(q, rs);

for i:= 0  step  1  until  (NS-2)/2  do

begin for k:= 1  step  1  until  2  do

        P[0, 1, i, k] := I21[2×i-1, k] ×(-1)↑ i;

        C. F. (P[0, 1, i, 1] ,  P[0, 1, i, 2] );

end i ;

for i:= 0  step  1  until  NS/2  do

begin for k:= 1  step  1  until  2  do

        R[0, 1, i, k] := I21[ 2×i,  k] ×(-1) ↑ i;

end  i;

I21[0, 1] := NS/2;

I22[0, 1] := (NS-2)/2;
```

comfact (R, 0, 1, I21, P, 0, 1, I22) results: (as, NS);

<u>comment</u>: the array "as" contains the common factor of $P_1(q, rs)$ and $R_1(q, rs)$. We will now compute an approximate complex root of the common factor. ;

<u>comment</u>: set all switches to return to b2;

ss[1] :=3 ;

ss[2] := 3 ;

ss[4] := 3 ;

ss3:= ss[3] ;

<u>go</u> <u>to</u> st;

b2: <u>comment</u>: b contains an approximate complex root of the common factor of $P_1(q, rs)$ and $R_1(q, rs)$, hence

$$y = rs + \sqrt{-b}$$

is an approximate root of $f(y) = 0$. ;

<u>for</u> i:= 1 <u>step</u> 1 <u>until</u> 2 <u>do</u>

b[i] := -b[i] ;

csqrt (b) results: (b) ;

b[1] := rs[1] / rs[2] +b[1] ;

<u>go</u> <u>to</u> return (ss3);

c2: <u>comment</u>: $D_q(P_1, R_1) \neq 0$ and $N = 4$;

<u>comment</u>: save the coefficients of $R_1(q, h)$;

<u>for</u> i:= 0 <u>step</u> 1 <u>until</u> 2 <u>do</u>

```
     for k:= 0  step  1  until  4-2×i  do
     for m:= 1  step  1  until  2  do
     RS[i,k,m] := R[i,1,k,m] ;
     comment:  we will compute an approximate root of
     Dq(P1,R1) = 0. ;
     for i:= 0  step  1  until  6  do
     for k:= 1  step  1  until  2  do
     begin  m:= NS-1;
              as[i,k] := S[m,1,i,k] ;
     end;
     NS:= 6;
     comment:  set the return switches to   b3;
     ss[1] := 4
     ss[2] := 4;
     ss[3] := 4;
     ss4:= ss[4] ;
     go to st;
b3:  comment:  b   contains an  approximate complex root of
     Dq(P1,R1) = 0;
     comment:  we can now evaluate   P1(q,b)  and   R1(q,b);
     for i:= 0  step  1  until  2  do
     begin for k:= 0  step  1  until  4-2×i  do
              begin  T21[k,1] := RS[i,k,1]/RS[i,k,2] ;
```

```
                T21[k, 2] := 0;
            end ;
            m:= 4-2×i ;
            cf(T21, m, b) results: (T11);
            T21[i, 1] := T11[1] ;
            T21[i, 2] := T11[2] ;
        end ;
        for i:= 0 step 1 until 2 do
        for k:= 1 step 1 until 2 do
        as[ i, k] := T21[2×i, k] ×(-1)↑ i;
        case ii (as) results: (T11, T12);
        m:= 2;
        cf(as, m, T11) results (T13);
        if T13[1] < .0001 then go to c5 else
        if T13[2] < .0001 then go to c5;
        T12[1] := T11[1] ;
        T12[2] ;= T11[2] ;
    c5: T12[1] := -T12[1] ;
        T12[2] := -T12[1] ;
        csqrt (T12) results: (T12);
        b[1] := b[1] +T12[1] ;
        b[2] ;= b[2] +T12[2] ;
        go to return (ss4);
```

```
M2: comment: the method that was developed in Chapter II does
    not apply.  We will try the method that was developed in
    Chapter III;
    comment: compute the number of factors of  2  in  N;
    t:= 0 ;
M3: i:= NS-2x entier (NS/2);
    if i:= 0  then
    begin  t:= t+1;
            NS:= NS/2;
            go to M3;
    end ;
    NS:= N;
    for i:= 1  step  1  until  t  do
    begin cees (as, NS) results:  (c);
            form PR(C,NS) results: (P,R)
            comment:  we will save the coefficients in the arrays
            P  and  R  for later use. ;
            for k:= 0  step  1  until  (NS-2)/2 do
            for m:= 0  step  1  until  NS-2×k-1  do
            for s:= 1  step  1  until  2  do
            PS[i,k,m,s] := P[k, 1,m,s] ;
            for k:= 0  step  1  until  NS/2  do
            for m:= 0  step  1  until  NS-2×k  do
```

```
        for s:= 1 step 1 until 2 do
        RS[i,k,m,s] := R[k,1,m,s] ;
        m:= NS/2;
        s:= (NS-2)/2;
        k:= NS×(NS-1)/2;
        comment: we will form Sylvester's eliminant;
        Sylvester (R,P,m,s,k) results: (S,I21);
        for k:= 0 step 1 until NS-2 do
        begin m:= k+1;
                ratmult (S,k,I21,S,m,I21,S,m,I21);
        end ;
        m:= NS-1 ;
        comfact (S,m,1,I21,S,m,2,I21) results: (I22, k);
        m:= NS×(NS-1)/2;
        for k:= 0 step 1 until m do
        for s:= 1 step 1 until 2 do
        as[k,m] := s[m,1,k,s] ;
        I11[i] := NS ;
        NS:= m;
end;
comment: from Theorem VII of Chapter III, the degree of
Sylvester's eliminant is odd. Therefore, we will compute
an approximate real root of Sylvester's eliminant. ;
```

```
realroot (eps,f,as,NS) results  (I12);
b[1] := I12[1]/I12[2] ;
b[2] := 0;
comment: we can now proceed as in Theorem VIII of Chapter
III;
for i:= 1 step 1 until t do
begin s:= I11[i] ;
      for k:= 0 step 1 until (s-2)/2 do
      begin m:= 0 step 1 until s-2×k-1 do
            begin T21[m,1] := PS[i,k,m,1]/PS[i,k,m,2] ;
                  T21[m,2] :=0;
            end;
      m:= s-2×k-1 ;
      cf (T21,m,b) results: (T12) ;
      T21[k,1] := T12[1] ;
      T21[k,2] := T12[2] ;
      end;
      for k:= 0 step 1 until s/2 do
      begin for m:= 0 step 1 until s-2×k do
            begin T22[k,1]:=RS[i,k,m,1]/RS[i,k,m,2] ;
                  T22[k,2] := 0;
            end;
            m:= s-2×k;
```

```
            cf(T22, m, b)  results:  (T12);
            T22[k, 1] := T12[1] ;
            T22[k, 2] := T12[2] ;
    end ;
    m:= s/2 ;
    k:= (s-2)/2;
    complex cf(T22, m, T21, k)  results:  (T23);
    T23[0, 1] := -T23[0, 1] ;
    T23[0, 2] := -T23[0, 2] ;
    for k:= 1  step  1  until  2  do
            begin  T12[k] := T23[0, k] ;
                   T13[k] := T23[1, k] ;
            end;
            cdiv (T12, T13) results:  (T12);
            for k:= 1  step  1  until  2  do
            T12[k] := -T12[k] ;
            csqrt(T12) results:  T (12) ;
            for k:= 1  step  1  until  2  do
            b[k] := b[k] +T12[k]  ;
    end  i ;
stop:
end comroot;
```

```
procedure cf(a, N, z) results: (b);
real arrays a, b; integer N;
comment: the procedure cf will evaluate a function f(y) with
complex coefficients.
```

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

The values for the formal parameter a are stored in the array a[0:n, 1:2]. The values 0 to N for the first subscript refer to the coefficient a_i. The values 1 and 2 for the second subscript refer to the real and imaginary part respectively of the coefficient a_i specified by the first subscript. The formal parameter b stands for the value of the function evaluated for the complex value z. The value for b is stored in the array b[1:2]. The value for z is stored in the array z[1:2]. The values 1 and 2 for the subscript refer to the real and imaginary part respectively of the arrays b and z.

```
begin integer i, k, m;   real array T[1:2];
        for i:= 1 step 1 until 2 do
        b[1] := a[N, i];
        for i:= 0 step 1 until N-1 do
        begin cmult (b, z) results: (T);
                for k:= 1 step 1 until 2 do
                b[k] := T[k] + a[i, k];
```

```
        end;

end;
```

<u>procedure</u> cees (a, N) results: (c) ;

<u>integer</u> <u>array</u> a, c; <u>integer</u> N;

comment: the procedure cees will compute

$$c_i(h) \quad (i = 0 \text{ to } N) ,$$

where

$$c_i(h) = \frac{f^i(h)}{i!} \quad ,$$

and

$$f(h) = \sum_{i=0}^{n} a_i h^i \quad .$$

the values for the formal parameter a are stored in the array a[0:N, 1:2] . The values 0 to N for the first subscript refer to the coefficient a_i of h^i. Each coefficient specified by the first subscript is expressed as the quotient of two integers. The values 1 and 2 for the second subscript refer to the numerator and denominator respectively of this quotient. The values for the formal parameter c are stored in the array c[0:N, 0:N, 1:2] . The values 0 to N for the first subscript refer to the polynomial $c_i(h)$. The values 0 to N for the second subscript refer to the coefficients of the polynomial in h denoted by the first subscript. The values 1 and 2 for the third subscript refer to the numerator and denominator respectively of the coefficients referred to by subscripts one and two. ;

```
begin integer i, k; integer array P[0:N, 0:N] ;
        P[0,0] := 1;
        P[1,0] := 1;
        P[1,1] := 1;
        for k:= 2 step 1 until N do
        for i:= 2 step 1 until k-2 do
        P[k,i] := P[k-1, i-1] + P[k-1, i] ;
        for k:= 0 step 1 until N do
        for i:= 0 step 1 until N-k do
        begin c[k,i,1] := P[i,k] X a[N-i, 1] ;
                c[k,i,2] := a[N-i,2] ;
                C.F. (c[k,i,1] , c[k,i,2])
        end ;
end cees
```

<u>procedure</u> realroot (eps, f, a, N) results: (b);

<u>real</u> eps;

<u>integer</u> <u>array</u> a, b;

<u>real</u> <u>procedure</u> f;

<u>integer</u> N;

<u>comment:</u> the procedure realroot will compute an approximate real root of the equation

$$f(z) = 0,$$

where

$$f(z) = \sum_{i=0}^{n} a_i z^i .$$

The values for the formal parameter a are stored in the integer array a[0:N, 1:2]. The values $i(i = 0, 1, \cdots N)$ for the first subscript of the array a refer to the coefficient a_i of z^i. The coefficients a_i are rational numbers and are stored as ordered pairs of integers. The values 1 and 2 for the second subscript refer to the numerator and denominator respectively of the coefficient specified by the first subscript. The formal parameter b stands for the approximate root of $f(z) = 0$. The value for b is stored in the integer array b[1:2]. The values 1 and 2 for the first subscript of the array b. refer to the numerator and denominator respectively. The approximate root b must be accurate within the given tolerance eps.;

```
begin integer array Fz0[1:2] , Fz1[1:2] ,
    Fz2[1:2] ,  F2[1:2] ,  z0[1:2] ,  Fz1[1:2]
    integer  i, k, m ;
    z0[1] := 0 ;
    z1[1] := 1 ;
    z1[2] := 1 ;
    f(a, N, z0) results: (Fz0) ;
    for i:= 1  step  1  until  5  do
    begin for  k:= 1  step  1  until  2  do
        begin f(a,N,z1) results:  (Fz1);
            if  (Fz0[1]/Fz0[2]) (Fz1[1]/Fz1[2]) 0
            then go to  R1;
            z1[1] := -z1[1] ;
        end;
        z1[1] := z1[1] × 10↑ i;
        C.F. (z1[1] , z1[2]);
    end;
    comment:  we cannot find two values  z0  and  z1  such that
    f(z0)  f(z1) < 0 ;
    go to error;
R1: if (z0[1]/z0[2]) (z1[1]/z1[2])  then
    z2[1] := z1[1] × z0[2] - z1[2] × z0[1]  else
    z2[1] := z0[1] × z1[2] - z0[2] × z1[1]  ;
    z2[2] := 2 × z1[2] × z0[2] ;
```

```
        C.F. (z2[1] , z2[2]);
        f(a,N,z2) results: (Fz2);
        if abs (Fz2[1]/Fz2[2])< eps then
        begin
                for i:= 1 step 1 until 2 do
                        b[i] := F2[i];
                go to R2;
        end; else
        if (Fz2[1]/ Fz2[2] )×(Fz1[1] /Fz1[2] ) < 0
        then for i := 1 step 1 until 2 do
        Fz0[i] := Fz2[i] else
        for i:= 1 step 1 until 2 do
        Fz1[i] := Fz2[i] ;
        go to R1;
R2:
end realroot;
```

```
procedure f(c,N,z) results: (y);
integer array c,z,y; integer N;
comment: procedure f will compute a value for y where y is
defined as follows:
```

$$y = \sum_{i=0}^{n} c_i z^i .$$

The values for the formal parameter c are stored in the integer array c[0:N, 1:2] . The first subscript with values from 0 to N refer to the coefficient of z^i. The second subscript with values 1 and 2 refers to the numerator and denominator respectively of the coefficient of z^i. The formal parameters z and y are stored in arrays z[1:2] and y[1:2] respectively. The subscript with values 1 and 2 refers to the numerator or denominator respectively of the values for y and z.

```
begin integer i,k;
      for i:= 1 step 1 until 2 do
      y[i] := c[N,i] ;
      for i:= 0 step 1 until N-1 do
      begin for k: = 1 step 1 until 2 do
            y[k] := z[k]×y[k] + c[N-i-1, k] ;
            C.F.(y[1] ; y[2] );
      end k;
end f;
```

```
procedure caseii(a) results: (b1, b2);
real array a, b1, b2;
comment: the procedure caseii will compute two approximate
complex roots of an algebraic equation f(y) = 0, where the coeffi-
cients of f(y) are complex and the degree of f(y) is 2. The
values for the formal parameter a are stored in the array
a[0:2, 1:2]. The values 0 to 2 for the first subscript refer to the
complex coefficients of f(y). The values 1 and 2 for the second
subscript refer to the real and imaginary part respectively of the
complex coefficient designated by the first subscript. The formal
parameters b1 and b2 stand for the two approximate roots of
f(y) = 0. The values for these roots are stored in the arrays
b1[1:2], and b2[1:2]. The values 1 and 2 for the subscript
refer to the real or imaginary part respectively of the complex root.
begin integer i; real array T0[1:2], T1[1:2], T2[1:2],
        T3[1:2], T4[1:2] ;
        for i:= 1 step 1 until 2 do
        begin T0[i] := a[0, 1] ;
                T1[i] := a[1, i] ;
                T2[i] := a[2, i] ;
        end i;
        cmult (T1, T1) results: (T3);
        cmult (T0, T2) results: (T4);
```

```
        for i:= 1 step 1 until 2 do
        T3[i] := T3[i] -4×T4[i] ;
        csqrt (T3) results: (T4);
        for i:= 1 step 1 until 2 do
        begin T3[i] := -T1[i] + T4[i] ;
              T0[i] := 2× T2[i] ;
              T1[i] := -T1[i] -T4[ i] ;
        end i;
        cdiv (T3,T0) results: (b1);
        cdiv (T1,T0) results: (b2);
end caseii
```

<u>procedure</u> form PR(c, N) results: (P,R);

<u>integer</u> <u>array</u> c, P, R; <u>integer</u> N;

<u>comment</u>: the procedure form PR will form the coefficients of two polynomials P(q,h) and R(q,h).

$$P(q,h) = c_1(h) - c_3(h)q \cdots (-1)^{\frac{n-2}{2}} c_{n-1} q^{\frac{n-2}{2}}$$

$$R(q,h) = c_0(h) - c_2(h)_q \cdots (-1)^{\frac{n-2}{2}} c_{n-2} q^{\frac{n-2}{2}} + (-1)^{\frac{n}{2}} c_n q^{\frac{n}{2}} .$$

The values for the formal parameter c are stored in the array c[0:N, 0:N, 1:2] . T he values in the array c are the results of the procedure cees. The values for the formal parameters P and R are stored in the arrays P[0:(N-2)/2, 1:2, 0:N, 1:2] , and R[0:N/2, 1:2, 0:N, 1:2] respectively. The values 0 to (N-2)/2 for the first subscript in the array P refer to the polynomial in h that is a coefficient of q^i. The values 0 to N for the second subscript refer to the coefficients of the polynomial in h that was specified by the first subscript. The values 1 and 2 for the third subscript refer to the numerator and denominator respectively of the coefficient specified by the first two subscripts. Similarly, the values 0 to N/2 for the first subscript of the array R refer to the polynomial in h that is a coefficient of q^i. The second and third subscripts for the array R have the same meaning as the second and their subscripts for the array P respectively.

```
begin integer  i, k,  L;
      for  i: = 0  step  1  until  (N-2)/2  do
      for  k:= 0  step  1  until  N-2×i-1  do
      begin for  L:= 1  step  1  until  2  do
            P[i, 1, k, L] := c[2×i-1, k, L]  (-1)↑i;
            C. F.  (P[ i, 1, k, 1] ,  P[i, 1, k, 2] );
      end  k;
      for  i:= 0  step  1  until  N/2  do
      for  k: = 0  step  1  until  N-2 i  do
      begin for  L:= 1  step  1  until  2  do
            R[i, 1, k, L] := c[2×i, k, L] ×(-1)↑i;
            C. F.  (R[ i, 1, k, 1] ,  R[i, 1, k, 2] ;
      end  k;
end form PR ;
```

```
procedure csqrt (a) results: (b);
real array a,b;
comment: the procedure csqrt computes the square root of the
complex number a and stores the result into b. The values for
the formal parameters a and b are stored in the arrays a[1:2],
and b[1:2] respectively. The values 1 and 2 for the sub-
script in the arrays a and b refer to the real and imaginary
part respectively of the complex number.;
begin real z, T1, PI;
        PI:= 3.1416
        z:= sqrt(sqrt(a[1]↑2+a[2]↑2));
        T1:= arctan(a[2]/a[1])/2;
        if a[1] < 0 then T1:= T1+PI;
        b[1] := z×cos(T1);
        b[2] := z×sin(T1);
end csqrt ;
```

```
procedure cdiv (a,b)results: (c);
real array  a,b,c;
comment: the procedure  cdiv  divides the complex number  a  by
the complex number  b  and stores the quotient into  c.  The values
for the formal parameters  a,b, and c  are stored in the arrays
a[1:2], b[1:2], and c[1:2]  respectively. The values  1  and  2
for the subscript in the arrays  a, b, and c  refer to the real and
imaginary part respectively of the complex number;
begin real  z, T1, T2, T3, PI;
        PI:= 3.1416
        z:= sqrt(a[1]↑2+a[2]↑2)/sqrt(b[1]↑2+b[2]↑2);
        T1:= arctan(a[2]/a[1]);
        T2:= arctan(b[2]/b[1]);
        if a[1] < 0 then T1:= T1 + PI;
        if b[1] < 0 then T2:= T2 + PI;
        T3:= T1-T2;
        c[1] := z×cos(T3);
        c[2] := z×sin(T3);
end cdiv ;
```

```
procedure cmult (a,b) results: (c);
real array a, b, c;
comment: the procedure cmult multiplies two complex numbers
a and b together and stores the product into c. The values for
the formal parameters a, b, and c are stored in the arrays
a[1:2] , b[1:2] , and c[1:2] respectively. The values 1 and 2
for the subscript in the arrays a, b, and c refer to the real and
imaginary part respectively of the complex number.
begin integer i ;
        c[1] := a[1]×b[1] -a[2]×b[2] ;
        c[2] := a[2]×b[1] +a[1]×b[2] ;
end cmult ;
```

<u>procedure</u> complex cf (a, n, b, m) results: (c);

<u>array</u> a, b, c;

<u>integer</u> n, m;

<u>comment</u>: the procedure complex cf will compute the common factor of two polynomials f(y) and g(y) with complex coefficients. The common factor is assumed to be linear.

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

and

$$g(y) = \sum_{k=0}^{m} b_k y^k \ .$$

The values for the formal parameters a and b are stored in the arrays a[0:n, 1:2] and b[0:m, 1:2] respectively. The values i(i=0, 1, ⋯ n) and k(k=0, 1, ⋯ m) for the first subscript of the arrays a and b respectively, refer to the coefficients a and b respectively. The coefficients a and b are complex numbers and are stored in ordered pairs of real numbers. The values 1 and 2 for the second subscript refer to the real and imaginary part respectively of the coefficient that was specified by the first subscript. The formal parameter c stands for the linear common factor of f(y) and g(y). The values for c are stored in the array c[0:1, 1:2]. The values 0 and 1 for the first subscript refer to the coefficient c_0 and c_1 respectively, where $c(y) = c_1 y + c_0$. The values 1 and 2 for the second subscript refer to the real and imaginary part respectively of the coefficient referred to be the first coefficient. The method of computation is based on the Euclidean algorithm. ;

```
begin integer i, k, s, p, NS, ms
    array R[0:m, 1:2] , as[0:n, 1:2] , bs[0:m, 1:2] ;
    NS:= n; ms:= m;
    for i:= 0 step 1 until n do
    for k:= 0 step 1 until 2 do
    as [i,k] := a[i,k] ;
    for i:= 0 step 1 until m do
    for k:= 1 step 1 until 2 do
    bs [i,k] := b[i,k] ;
    for i:= 0 step 1 until m do
    begin polydiv (as, Ns, bs, ms) results (R,p);
        for k:= 0 step 1 until ms do
        for s:= 1 step 1 until p do
        as [k, 2] := bs[k, 2] ;
        for k:= 0 step 1 until p do
        for s:= 1 step 1 until 2 do
        bs[k, 2] := R[k, s] ;
        NS:= ms;
```

```
            ms:= p ;
        end ;
        for i:= 0 step 1 until 1 do
        for k: = 1 step 1 until 2 do
        c[i, k] := R[i, k] ;
    end complex cf ;
```

<u>procedure</u> polydiv (a, n, b, m) results: (R, p);

<u>array</u> a, b, R;

<u>integer</u> m, n, p;

<u>comment</u>: the procedure polydiv computes the remainder when polynomial f(y) is divided by polynomial g(y).

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

and

$$g(y) = \sum_{k=0}^{m} b_k y^k .$$

The values for the formal parameters a and b are stored in the arrays a[0:n, 1:2] and b[0:m, 1:2] respectively. The values $i(i=0, 1, \cdots n)$ and $k(k=0, 1, \cdots m)$ for the first subscript of the arrays a and b respectively, refer to the coefficients a_i and b_k respectively. The coefficients a_i and b_k are complex numbers and are stored in ordered pairs of real numbers. The values 1 and 2 for the second subscript refer to the real and imaginary part respectively of the coefficient that was specified by the first subscript. The formal parameter R stands for the remainder polynomial. The values for R are stored in the array R[0:m, 1:2]. The values $(0, 1, 2, \cdots m-1)$ for the first subscript refer to the coefficients of the remainder polynomial. The values 1 and 2 for the second subscript refer to the real and imaginary part respectively of the coefficients referred to by the first subscript. The integer p contains the degree of the remainder polynomial. n > m;

```
begin integer i, k, s;
      array Q[0:n-m, 1:2] , T1[1:2] , T2[1:2] , as[0:n, 1:2] ;
      for i:= 0 step 1 until n do
      for k: = 1 step 1 until 2 do
      as[i, k] := a[i, k] ;
      for i:= n-m step -1 until 0 do
      begin
            for k: = 1 step 1 until 2 do
            begin T1[k] := as[n, k] ;
                  T2[k] := b[ m, k] ;
            end ;
            cdiv (T1, T2) results: (T1) ;
            for k:= 0 step 1 until m-1 do
            begin
                  for s:= 1 step 1 until 2 do
                  T2[s]:= b[m-k-1, s] ;
                  cmult (T1, T2) results: (T1) ;
                  for s:= 1 step 1 until 2 do
                  as[n-k, s] := as[n-k-1, s] ;
                  for s:= m step 1 until n-1 do
                  begin as[n-s, 1] := as[n-s-1, 1] ;
```

```
                    as[n-s, 2] := as[n-s-1, 2] ;
               end s ;
          end k ;
     end i ;
     for i:= 0 step 1 until m-1 do
     begin R[i, 1] := as[n-m+1+i, 1] ;
          R[i, 2]:=as[ n-m+1+i, 2] ;
     end ;
end polydiv ;
```

APPENDIX B

The procedures in Appendix B were copied from a thesis written by John Ries Holroyd, Oregon State University, 1962.

```
procedure ratmult (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);
        comfact (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 ratmult ;
```

```
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 to a real parameter which may
have value 1 or 2 as to numerator or denominator of the poly-
nomial 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] X J[j, v, m, 1] ;
            f:= I[i, u, t, 2] X J[j, v, m, 2] ;
            e:= exH[ m+t, 2] + fxH[m+t, 1] ;
            f:= fxH[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 comfact (I, i, u, L, J, j, v, M),   results: (E, 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.

$$a = ca'$$

$$b = cb'$$

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 algortihm 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.

$$b = a \cdot q[1] + r[1] \qquad (1)$$

$$a = r[1]\, q[2] + r[2] \qquad (2)$$

$$r[1] = r[2]\, q[3] + r[3] \qquad (3)$$

. . . .

$$r[n-1] = r[n]\, q[n+1] + r[n+1] \qquad (n)$$

$$r[n] = r[n+1]\, q[n+2] \qquad (n+1)$$

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 C 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;
H3: for t:= 0 step 1 until s do
    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 comfact ;
```

```
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 cor-
responding 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-hx entier (g/h) );
      if r=0 then go to H2 ;
      g:= h ;
      h:= r ;
      go to H1 ;
  H2: a:= a/h ;
      b:= b/h ;
end C.F. ;
```

<u>procedure</u> Sylvester (A, B, m, s, r) results: (C, e) ;

<u>comment:</u> the procedure Sylvester is a recursive procedure for the reduction of Sylvester's eliminant for the polynomials A and B. The values for the formal parameters A and B are stored in the arrays A[0:n, 1:2, 0:r, 1:2] and B[0:n, 1:2, 0:r, 1:2] respectively. The values 0 to n for the first subscript refer to the coefficients a_i and b_i for the polynomials A and B respectively. The coefficients a_i and b_i are the quqtient of two polynomials. The values 1 and 2 for the second subscript refer to the numerator and denominator respectively of the rational polynomials a and b. The values 0 to r for the third subscript refer to the coefficients of the polynomials that are specified by the first two subscripts. The values 1 and 2 for the fourth subscript refer to the numerator or denominator respectively of the rational number that is specified by the first three coefficients. The Sylvester's eliminant is the product of all polynomials stored in the array C. The degree of the polynomials whose coefficients are stored in the array C are stored in the array e[0:n, 1:2] . The values 0 to n for the first subscript refer to the polynomials stored in the array C. The values 1 and 2 for the second subscript refer to the numerator and denominator respectively of the rational polynomial that is specified by the first subscript. The actual procedure is not included because of its excessive length.

APPENDIX C

EXAMPLE

We will compute a root of $f(y)$, where

$$f(y) = y^6 - 24y^3 + 196y^2 - 336y + 288.$$

Using the division algorithm, dividing $f(y)$ by (y^2+q) we get,

$$f(y) = Q(y,q)(y^2+q) + P(q)y + R(q) .$$

$$P(q) = 24q - 336 ,$$

and

$$R(q) = q^2 - 196q + 288 .$$

We can test Sylvester's eliminant $D_q(P,R)$ to determine if the polynomials $P(q)$ and $R(q)$ have a common factor

$$D_q(P,R) = \begin{vmatrix} -1 & 0 & -196 & 288 & 0 \\ 0 & -1 & 0 & -196 & 288 \\ 0 & 24 & -336 & 0 & 0 \\ 0 & 0 & 24 & -336 & 0 \\ 0 & 0 & 0 & 24 & -336 \end{vmatrix} \qquad (1)$$

From evaluation of (1) we get

$$D_q(P,R) = -71884800 .$$

$D_q(P,R) \neq 0$, hence the polynomials $P(q)$ and $R(q)$ do not have a common factor.

By expanding $f(y)$ we get,

$$f(y) = c_6(y-h)^6 + c_5(y-h)^5 + c_4(y-h)^4 + c_3(y-h)^3 + c_2(y-h)^2 + c_1(x-h) + c_0.$$

$$c_0 = h^6 - 24h^3 + 196h^2 - 336h + 288.$$

$$c_1 = 6h^5 - 72h^2 + 392h - 336.$$

$$c_2 = 15h^4 - 72h + 196 .$$

$$c_3 = 20h^3 - 24.$$

$$c_4 = 15h^2 .$$

$$c_5 = 6h .$$

$$c_6 = 1 .$$

Using the division algorithm, dividing $f(y)$ by $((y-h)^2 + q)$ we get,

$$f(y) = Q(q,y,h)\,((y-h)^2 + q) + P_1(q,h)\,(y-h) + R(q,h) .$$

$$P_1(q,h) = c_5q^2 - c_3q + c_1,$$

and

$$R_1(q,h) = -c_6q^3 + c_4q^2 - c_2q + c_0 .$$

$$D_q(P_1,R_1) = \begin{vmatrix} -c_6 & c_4 & -c_2 & c_0 & 0 \\ 0 & -c_6 & c_4 & -c_2 & c_0 \\ c_5 & -c_3 & c_1 & 0 & 0 \\ 0 & c_5 & -c_3 & c_1 & 0 \\ 0 & 0 & c_5 & -c_3 & c_1 \end{vmatrix} \qquad (2)$$

From (2) we get,

$$\begin{aligned} D_q(P_1,R_1) = {} & k\cdot(h-1)\cdot(h-2)\cdot(h+3)\cdot(h-\tfrac{3}{2}-\tfrac{3}{2}i)\cdot(h-\tfrac{3}{2}+\tfrac{3}{2}i)\cdot \\ & (h-1-2i)\cdot(h-1+2i)\cdot(h+1+i)\cdot(h+1-i)\cdot(h+\tfrac{1}{2}-\tfrac{5}{2}i) \\ & (h+\tfrac{1}{2}+\tfrac{5}{2}i)\cdot(h+\tfrac{1}{2}+\tfrac{i}{2})\ (h+\tfrac{1}{2}-\tfrac{i}{2})\ . \end{aligned}$$

$D_q(P_1,R_1)$ is a polynomial in h that has a degree of 15. Therefore, we can compute a real root α of $D_q(P_1,R_1)$. If $\alpha = 1$, then

$$P_1(q,1) = 6q^2 + 4q - 10 ,$$

and

$$R_1(q,1) = -q^3 + 15q^2 - 139q + 125.$$

Using the Euclidean algorithm to compute the common factor of

$P_1(q,1)$ and $R_1(q,1)$, we get

$$9(-q^3+15q^2-139q+125) = (-3q+47)\cdot(6q^2+4q-10)/2+1360(-q+1),$$

$$(6q^2+4q-10)/2 = (-q+1)\cdot(-3q-5) .$$

Hence $(-q+1)$ is a common factor of $P_1(q,1)$ and $R_1(q,1)$. Therefore $P_1(1,1) = R_1(1,1) = 0$, thus making $y = 1\pm i$ roots of $f(y) = 0$.

If, when we computed a real root of $D_q(P_1, R_1)$, we had found that $\alpha = 2$, then

$$P_1(q,2) = 12q^2 - 136q + 352 ,$$

and

$$R_1(q,2) = -q^3 + 60q^2 - 292q + 272 .$$

Using the Euclidean algorithm to compute a common factor, we get

$$9(-q^3+60q^2-292q+272) = (-3q+146)\cdot(3q^2-34q+88)+2600(q-4),$$

$$3q^2-34q+88 = (3q-22)\cdot(q-4) .$$

Hence $q-4$ is the common factor of $P_1(q,2)$ and $R_1(q,2)$. Therefore $P_1(4,2) = R_1(4,2) = 0$, thus making $y = 2\pm 2i$ roots of $f(y) = 0$. If we find $\alpha = -3$ for a root of $D_q(P_1, R_1) = 0$, then

$$P_1(q, -3) = -18q^2 + 564q - 3618,$$

$$R_1(q, -3) = -q^3 + 135q^2 - 1627q + 4437 .$$

Using the Euclidean algorithm to compute the common factor of $P_1(q, -3)$ and $R_1(q, -3)$, we get

$$9(-q^3 + 135q^2 - 1627q + 4437) = (9q-933)\cdot(-18q^2+546q-3618)/2$$
$$+ (q-9))\cdot(147570) ,$$

$$(-18q^2+564q-3618)/2 = (q-9)\cdot(-9q+201) .$$

Hence $q-9$ is a common factor of $P_1(q, -3)$ and $R_1(q, -3)$. Therefore $P_1(9, -3) = R_1(9, -3) = 0$, thus making $y = -3 \pm i3$ roots of $f(y) = 0$.