An Electronic Digital Polynomial Root Extractor

R. R. JOHNSON†

Summary—Many mathematical techniques exist for factoring algebraic polynomials. Most require much computation and programming and are practical only for large machine computers. A different approach to the general problem is described. The mathematical method is an adaptation of a Taylor series approximation used to connect the problem and its formulation with a special machine implementation. The result is a simple digital computer capable of extracting the complex roots of an $n$th degree polynomial.

INTRODUCTION

The advent of the large-scale digital computer has resulted in a general emphasis on numerical methods. This emphasis has led to many applications of digital techniques to problems having special characteristics; computers designed to capitalize on these characteristics can obtain solutions rapidly and with considerable savings in computing equipment. Reductions in preparation and programming time can result from machines designed to handle problems appearing repeatedly in practice.

In scientific computations, one universally belabored problem is that of factoring algebraic polynomials. Many analog devices and many mathematical methods have been developed to solve this problem.

† California Institute of Technology, Pasadena, Calif.


nearest root. When it reaches a minimum value for the polynomial, the normal δ selection causes the computer to encircle the point of minimum absolute value.

The single memory channel contains 19 word positions. The real and imaginary components of the polynomial are in the first word; in each of the following 16 words is located the corresponding derivative of the polynomial (see Fig. 1).

These derivatives, evaluated at some convenient point such as the origin, comprise the initial input data. The computational principle is to evaluate each derivative at an incremental distance δ away from the initial co-ordinate (see Fig. 2). This approximation is the first term of a Taylor series expansion of the polynomial. All derivatives are recomputed for each step δ using this principle.

For the initial root location an improved approximation formula is used with a larger value of δ. Higher-order terms could be taken from the Taylor series, but a more elegant technique is to use part of each newly computed derivative:

\[
f(z + \Delta) = f(z) + \frac{\Delta}{2} f'(z) + \frac{\Delta^2}{3} f''(z) + \Delta^3 f'''(z + \Delta).
\]

The selection of δ is that which will assure |

\[ f(z + \Delta) \]

\[
\leq |f(z)|. \tag{3}
\]

For example, if both \( u \) and \( v \) are positive and their first derivatives are positive, \( \Delta = -\delta \). However, if \( u \) and \( v \) are positive and their first derivatives differ in sign, \( \Delta = \pm j\delta \) is the proper selection.

A simple case illustrates the behavior of the direction control when \( z \) approximates a root to within an amount \( \delta \). Consider all signs initially plus. The sequence of decisions prescribed by the rules above forces the computer to encircle the root. The computer remains in this mode until forced in another direction.

**Problem Preparation**

There are two restrictions placed on the polynomial. It is necessary to convert the coefficients to binary numbers and to limit all numbers to absolute values less than 1.0. There are 30 significant binary places available for the real and imaginary components of each derivative. Numbers are absolute value with sign when positive and zero's complements with sign when negative (see Fig. 3).

To ensure that \(|z| \leq 1\) for all roots, it is necessary to compute the radius of the contour in the \( z \) plane which encloses all \( n \) zeros of the polynomial

\[
R > \frac{a_k}{a_n} - 1, \tag{6}
\]

where \(|a_k|\) is the largest coefficient in the polynomial. A new argument is defined:

\[
w = \frac{z}{R}, \tag{7}
\]

which transforms the polynomial into

\[
f(w) = a_n(R)^n w^n + \cdots + a_1(R) w + a_0. \tag{8}
\]
The computer obtains the roots of (8). The roots of the original polynomial are found using (7). To insure that the absolute value of the \( n \) derivatives never exceed 1.0, it is necessary to divide \( f(w) \) by the largest coefficient:

\[
F(w) = \frac{f(w)}{|2n!Rw|},
\]

(9)

\[
= A_n w^n + \cdots + A_1 w + A_0,
\]

(10)

where \( i \) is the index of the largest coefficient in (8).

These computations must be done manually before inserting the derivatives into the computer. This is done to retain the computational simplicity of the computer. No multiplications or divisions are provided internally.

**D. Accuracy**

Each step in the approximation of the function and its derivatives has a truncation error

\[
e = \sum_{i=m+3}^{n} \frac{\Delta^i}{2^i} (i - 2) f^{(i)}(z).
\]

(11)

This is the error in the approximation to the \( m \)th derivative. The error in the function itself is of the order of

\[
\frac{\Delta^2}{12} f^{(3)}(z).
\]

Two values of \( \delta \) are used: \( 2^{-10} \) and \( 2^{-20} \). The computation sequence is to locate the root with the larger and refine it with the smaller \( \delta \). With the coarse \( \delta \), the truncation error at each step is of the order of \( 2^{-24} \). The total error in \( 2^{10} \) steps is \( 2^{-24} \). The root is refined using approximation (2) with \( \delta = 2^{-20} \). In \( 2^{10} \) steps this produces an error of \( 2^{-32} \).

Roundoff errors may propagate to the twenty-first binary position in \( 2^{10} \) steps so that the polynomial, its derivatives, and the value of \( z \) may be considered accurate to the twentieth binary place. The smallest increments in \( z \) are \( \delta = 2^{-20} \).

To obtain full accuracy for all roots, it is necessary to normalize by (7) and locate the roots approximately. Full significance is obtained for the smaller roots by renormalizing the polynomial to an \( R \) just greater than the modulus of the desired root.

**Machine Design**

The computer has three modes of operation:

1. **Input**
2. **Computation** with \( \delta = 2^{-10} \) or \( \delta = 2^{-20} \),
3. **Direction decision**.

The operational aspects of these modes have been described. This section gives a general description of the techniques by which that functional behavior is accomplished. The detailed logical design will be available from the California Institute of Technology.

**Input**

To describe the input routine, it is necessary to understand some of the internal features of the computer. Input itself is so simple that it will be convenient to present a more detailed picture of the computer in this section as well.

During the first moments of the input phase the computer orients itself with respect to its internal timing. This feature arises from a general consideration of the computer configuration (see Fig. 4).

To minimize the number of reading and writing heads on the drum, it was necessary to use one circulating register. This implied a single clock channel with no reference to an origin pulse on the drum. Timing signals are required, however, to indicate certain word positions; without having the circulating register exactly one drum circumference in length it is necessary to generate certain timing markers inside the register itself.

In addition to the polynomial and its derivatives, the memory channel contains two words, \( "z" \) and \( "\Delta" \). \( "\Delta" \) holds a marker used to add the correct \( \Delta \) to \( "z" \). \( "z" \) holds the value of the complex argument \( z \) of the polynomial. The timing signals are those necessary to distinguish:

1. The words \( f(z) \), \( "\Delta" \), \( "z" \), and \( f^{(0)}(z) \),
2. The internal word structure: blank bits, sign bits, real or imaginary bits, and the position of the \( \delta \)th binary place.

Phase Control designates the first group of timing signals; the second group are general timing signals.

The blank bits between words may contain two markers. These markers are entered under the control of the general timing signals when the input mode switch is closed. One marker precedes the words holding \( f(z) \) and \( "z" \); it serves to synchronize the phase control. The other is a movable marker identifying any particular word.

The general timing signals originate in the clock channel. Originally the drum is uniformly divided into 1,344 positions. The clock channel is derived by occasionally omitting a single pulse (see Fig. 5). The clock pulses to operate the computer are obtained from a free-running phantastron synchronized by pulses from the clock channel. An omitted pulse is filled in for clock-pulse use by the phantastron. With one pulse omitted at a time, there is little drift. The missing pulse operates
the general timing flip-flops; they change state whenever a pulse is omitted from the clock channel. Synchronization of the general timing pattern with the drum pattern is self-controlled and may take several word times after the computer is turned on.

Fig. 5—One word of the pulse configuration on the clock channel.

The input bits are entered via two lever switches; each bit is placed in a flip-flop and the desired word circulates back to the memory through that flip-flop.

Computation

There are three operations in the computation mode:
1. Multiplication by $\frac{1}{2}\delta$,  
2. Inversion of the bit sequences as indicated by (5),  
3. Addition of the respective components.

Multiplication by $\frac{1}{2}\delta$ is done by shifting the reading heads. This is indicated in Fig. 4; it is also the reason for limiting the absolute values to 1.0.

Inversion merely exchanges the bipoled positions of the real and imaginary components when $\Delta$ changes from real to imaginary. Two flip-flops are used for the inversion, and two separate adders are used to obtain a compromise between numbers of flip-flops and numbers of diodes.

Subtraction or addition is governed by Direction Control. $\frac{1}{2}\delta f'(z)$ and $\frac{1}{2}\delta f'(z+\Delta)$ are always added, but their sum is added to or subtracted from $f(z)$. The simplest technique is to combine the conversion to zero's complements of the subtrahend with the sequence inversion.

Direction Control

This is the most complicated logical operation in the computer. The decision is based on a truth table of all sign conditions in (5). The logic is derived from a similar table based on the information available internally in the computer. That is, due to sequence inversion and zero's complementation in $D1$ and $D2$ (Fig. 6), it is necessary to consider the form in which the sign information is available. One additional limitation on the decision logic is that the modified sign bits are not all available during any one clock time.

Two flip-flops, $G$ and $H$, are used to perform the decision. After each circulation of the memory channel a new value for $\Delta$ is determined from the new signs of $f(z)$ and $f'(z)$ and from the original states of $G$ and $H$. Referring to Fig. 6 and using the additional notation

$$B = \text{odd-even clock timer},$$
$$M1 = \text{reading flip-flop for } f(z),$$
$$T0 = \text{sign bit times (the real sign is in } M1 \text{ during } B \cdot T0 = 1, \text{ and the imaginary sign is in } M1 \text{ during } B \cdot T0 = 1),$$

the input equations to $G$ and $H$ are written:

$$1G = (D1 \times D2 \times B) \cdot (D2 \times M1) \cdot T0$$
$$0G = 1G$$
$$1H = (D2 \times M1) \cdot T0$$
$$0H = 1H.$$

These equations are given to illustrate the form in which the logical design is obtained. The bar denotes complementation and the $(\times)$ denotes the exclusive "or" operator

$$D1 \times D2 = D1 \cdot D2 + D1 \cdot D2.$$

The + and the $\cdot$ symbols designate logical union and intersection, respectively. Eqs. (12) describe a flip-flop having the following behavior.

<table>
<thead>
<tr>
<th>Input:</th>
<th>Output:</th>
<th>$G_{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1G_n$</td>
<td>$0G_n$</td>
<td>$G_n$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>$G_n$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$G_n$</td>
</tr>
</tbody>
</table>

The output is delayed one clock interval with respect to the input; $n$ denotes the clock interval.

The decisions produced by (12) are not exactly correct. In a few cases $G$ and $H$ will select the correct real or imaginary $\Delta$ but will give it the wrong algebraic sign. The computer takes one step in the wrong direction and then recognizes its mistake and proceeds properly.

This is a situation where it is easiest to make an occasional wrong decision without hurting the functional operation of the device. Several additional diodes and an extra flip-flop would be required to make the correct decision at all times.

Physical Specifications

Two types of flip-flops are used. Those described by (13) are the first. The second kind are described by the middle two rows of (13). The second type introduces a delay and power amplification only (see Fig. 7).
Note that only two power supplies are required. These supply the flip-flops, diode matrices, and the read-write amplifiers. Approximately 100 watts is dissipated exclusive of heaters.

The basic clock frequency is 80.7 kc derived from 1,344 pulses on a 5-inch drum rotating at 3,600 rpm.

Mode selection is under the control of several switches on the control panel. They are used for starting, inserting the timing markers, input, Δ selection, forced direction control, locating the movable marker, and selecting any desired output (see Fig. 8).

**CONCLUSION**

Several observations result from the design and construction of the machine. It requires about half the equipment necessary for a medium-speed general-purpose computer of moderate memory capacity. Where there is need for its special purpose, the ready availability of the computer and its ease of programming make it a valuable scientific tool. The binary coding and bit by bit input and output are limitations, but it is still felt that the equipment savings warrant them. Even more equipment could be eliminated if certain other features of the present computer were omitted.

While the logical techniques developed for the root extractor are conceptually useful in other digital devices, the computer itself has several other uses.

The machine can be considered as a function generator in two variables in the sense that it evaluates a polynomial of high degree. Inputs are any desired \( x \) and \( y \). Internal programming is set to approach the \( x \) and \( y \) inputs rather than to diminish \( |f(z + \Delta)| \). Discontinuous functions could be generated if there were some particular behavior characteristic of a higher derivative. Input could be to the derivative word position in this case.

In the field of servomechanisms the computer could be used to obtain root locus plots. A curve-plottter would be attached to read the "\( z \)" word in this case, and some form of internal gating would be used to indicate that a root had been located.

**ACKNOWLEDGMENT**

The material described above covers a portion of the work performed at the California Institute of Technology by the author in partial fulfillment of the requirements for the Ph.D. degree in Electrical Engineering. It was done during periods when the author was the holder of a Howard Hughes Fellowship and an International Business Machines Corporation Fellowship in Electrical Engineering. Particular acknowledgment is due Dr. Stanley P. Frankel for his invaluable ideas and suggestions contributed during the development of the computer, and to Dr. G. D. McCann for his helpful assistance throughout the project's duration.
EXTENSIVE operating experience with the ORDVAC and the ILLIAC has demonstrated to members of the University of Illinois Computer Laboratory the advantages of computers which are direct-coupled and operate asynchronously. This paper describes a set of building block circuits, which is the first result of a program to develop transistor circuits which would be suitable for an asynchronous machine.

This paper will not go into the detailed mathematics of the circuits. Instead, its purpose will be to explain the design procedure that was used, and to describe briefly some of the results obtained with the circuits. A relatively complete discussion of the design of the flip-flop circuit will be used to illustrate the general technique, whose principle is undoubtedly familiar to many readers. Instead of studying the behavior of the circuit for the nominal values of the components, expressions are obtained for its behavior when every parameter value varies to some specified extent. The circuit is designed to operate properly with these extreme values; ordinarily a malfunction will then occur only if several components vary simultaneously by slightly more than the allowable amounts, or if a single parameter exceeds the allowable variation to a very great extent. (One example of the latter might be the sudden decrease of the current gain to zero when a point contact transistor burns out.)

Two fundamental characteristics of asynchronous computers are vital considerations in the design of such circuits. First, the binary signals in a direct-coupled computer are dc voltage levels rather than trains of pulses produced by a clock. Second, the individual switching operations, such as changing the state of a flip-flop, do not have any predetermined maximum duration. Instead, the machine waits until a signal has been produced, signifying the successful completion of the operation, before initiating the succeeding step. As a result, variations in the transient response of the individual circuits will not cause a malfunction. Because of these two factors, it is feasible to concentrate upon dc behavior when designing building block circuits for asynchronous operation.

The flip-flop circuit is shown in Fig. 1. Two point contact transistors are used, with their emitters connected together and returned through a bias resistor, $R'_b$, to a positive supply voltage, $V_{cc}$. To the best of the present writer's knowledge, this circuit was first proposed at the Air Force Cambridge Research Center. This circuit can have two stable states, in each of which one transistor is cut off, the other conducting.

Fig. 1—Flip-flop circuit with two point-contact transistors.

With appropriate values for $R_b$ and $R'_b$, the relation between $v_e$, the emitter-ground voltage, and $i_e$, the emitter current for either transistor, is the familiar $N$-curve, shown in Fig. 2. The $N$-curve represents an approximation to the nonlinear characteristics of the transistor in which the device parameters are considered to be constant throughout each of three operating regions. These regions are:

I. Cutoff Region: emitter current is zero or negative.

II. Active Region: normal transistor action takes place.

III. Saturation Region: collector-base voltage is too small to maintain transistor action.

Fig. 2—N-Curve for point-contact transistor.

Values for the $N$-curve which are critical for the design of the flip-flop circuit are the peak voltage, $v_{ep}$, the

---

From the collection of the Computer History Museum (www.computerhistory.org)