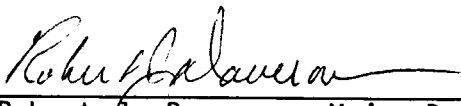
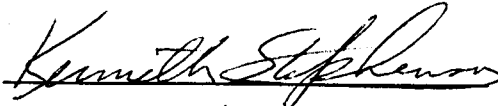
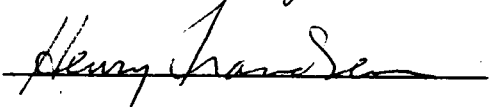


To the Graduate Council:

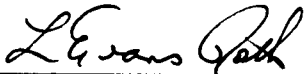
I am submitting herewith a thesis written by Wallace G. Hynds III entitled "A Computer Evaluation of Topological Knots." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mathematics.


Robert J. Daverman, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:


Vice Chancellor
Graduate Studies and Research

A COMPUTER EVALUATION
OF TOPOLOGICAL KNOTS

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Wallace G. Hynds III
December 1982

3065101

Copyrighted by Wallace G. Hynds III 1982

All Rights Reserved

ABSTRACT

The purpose of this thesis is to present an algorithm to aid knot theorists in the classification of topological knots. To make the algorithm useful and available to knot theorists, it is specifically designed to be programmed on a microcomputer. A program in Z-80 assembler to execute the algorithm is presented.

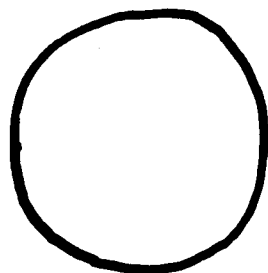
Presentations of knot groups are rather easy to extract by hand from a given knot. Free derivatives are used to convert the presentation of the knot group to a Jacobian matrix. A reduction to one variable of the determinant of this matrix yields the Alexander polynomial. This Alexander polynomial is a powerful classification tool with the added feature of ease of comparison.

TABLE OF CONTENTS

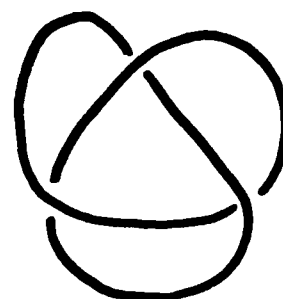
CHAPTER	PAGE
INTRODUCTION	1
I. KNOT THEORY	3
II. THE COMPUTER PROGRAM	14
LIST OF REFERENCES	26
APPENDICES	28
A. EXAMPLES	29
B. FLOWCHARTS	36
C. LISTING	39
VITA	56

INTRODUCTION

Consider these two views of a simple closed curve as a subset of $\mathbb{R}^3$.



THE TRIVIAL KNOT



THE TREFOIL

Since each is an image of S^1 , they are homeomorphic; however, they are obviously different in their placement in $\mathbb{R}^3$. We would like to form classification methods which would distinguish between curves placed differently. This forms a basis for Knot Theory and puts it in the family of placement problems. Since Knot Theory is one of the easiest placement problems to examine, it becomes a worthwhile subject for placement problems in general.

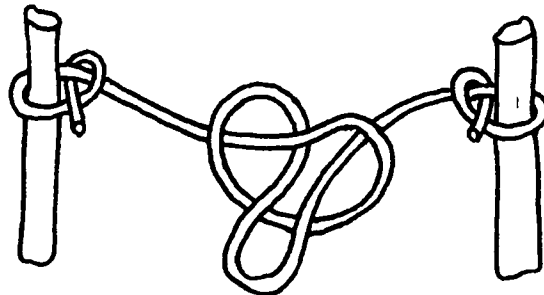
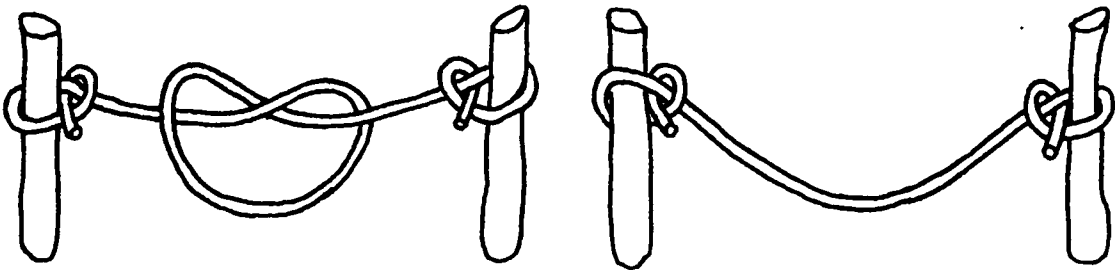
Reviewed here is a method for finding a presentation of a characteristic group of any tame knot. From that presentation a characteristic polynomial, called the Alexander polynomial, may be determined. An application of microcomputers is introduced by a program which executes the algorithm for computing the Alexander polynomial for such presentations of less than ten generators.

The program is written in Z-80 assembly language. Also discussed are aspects of this application which make it a suitable task for a microcomputer.

CHAPTER I

KNOT THEORY

The basic idea of Knot Theory may best be understood by examining a physical situation. If one knot may be rearranged into the form of another, then we say that these knots are equivalent. It is obvious that in the drawing below, knot (a) can only be twisted to the form of knot (b) by untying the ends. For this reason we say that knot (a) and knot (b) are not equivalent. It is again obvious that without untying the ends, knot (c) can be rearranged to the form of knot (b); hence we say that knot (c) and knot (b) are equivalent.



(c)

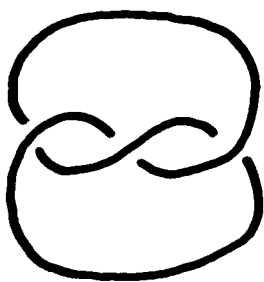
To facilitate some analysis of the situation, there is a need for some topological constructions which simulate the knot and the medium in which it exists. $\mathbb{R}^3$ is the obvious choice for the medium.

Some image of I^1 in $\mathbb{R}^3$ would seem to be a reasonable representation for the rope; however, the flexibility of topology essentially allows for the untying of the ends. This would be quite uninteresting, so we look to some other 1-manifolds.

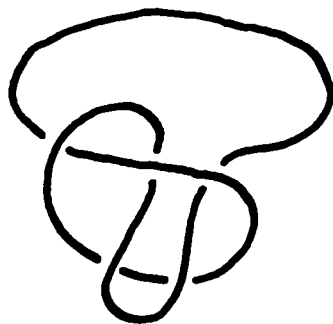
Some image of S^1 into $\mathbb{R}^3$ is a better choice. This fixes the ends of the rope together and robs topology of the power of untying the ends as above.

Hence we define a knot as the image of an embedding of S^1 into $\mathbb{R}^3$. This image is typically called K . Instead of focusing on the image of the embedding itself, we frequently concentrate on the geometric entity $\mathbb{R}^3 - K$ and we will call this the complementary space.

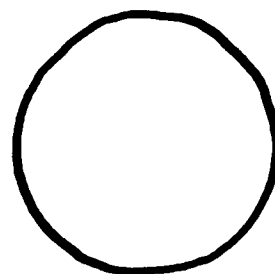
The drawing below shows the topological equivalents of the three knots in the drawing on the previous page.



(a)



(c)



(b)

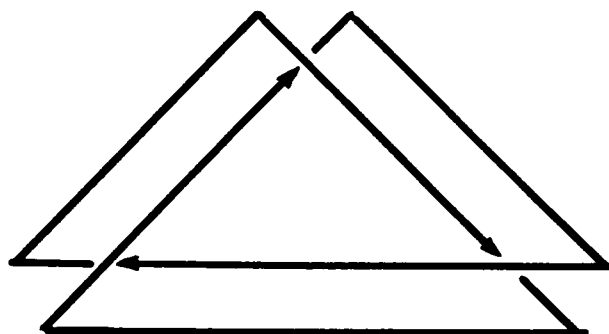
Equivalence among knots is defined by means of the homeomorphism guaranteed to exist between the images. Two knots, K_1 and K_2 , are said to be equivalent if and only if a homeomorphism from K_1 to K_2 can be extended to a homeomorphism of $\mathbb{R}^3$ to $\mathbb{R}^3$. A consequence of this definition is that equivalent knots have homeomorphic complementary spaces. The chief objective of the remainder of this chapter will be to select invariants which can be extracted from the complementary spaces. Spaces which possess identical invariants may or may not be equivalent, but those with distinct invariants can be identified as not equivalent.

Before we can attack invariants, we must eliminate from our collection of knots a family of undesirables known as wild knots. First, a polygonal knot is one which is composed of finitely many line segments connected end-to-end in a closed curve. A knot's equivalent which is polygonal is called its polygonal representation. Wild knots are those which have no polygonal representation. For such knots the invariants we want to extract may not exist. For this reason we will limit any further discussion to tame knots.

A handy device for classification of spaces is the fundamental group. The fundamental group of a knot's complementary space is called the knot group. Knot groups are by no means ideal for comparison purposes. They do, however, lead to other invariants and they are not too difficult to extract from a given knot.

To begin we need a special projection of the knot. Any polygonal knot can be arranged into what is called regular position with respect to a given plane in $\mathbb{R}^3$.

To be in regular position, two rules must hold. First, no more than two points in K can be projected to the same point in the plane. Second, no two points of K can be projected to the same point in the plane if either is a corner of the polygonal representation. Such a projection of the trefoil is drawn below.

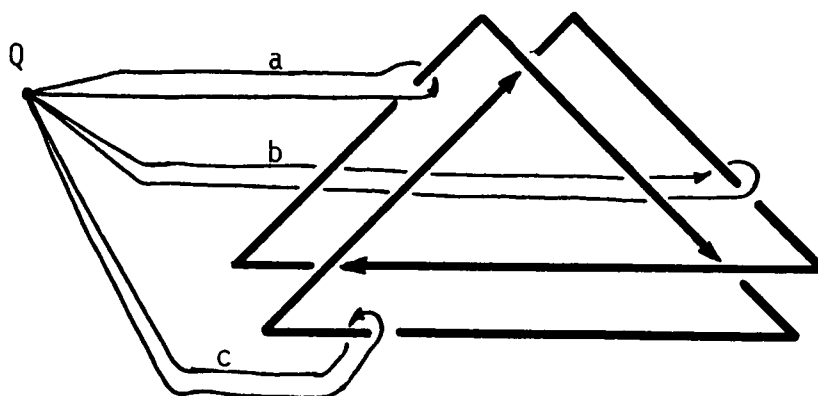


An arbitrary direction is assigned to K and transferred to the projection. This is indicated by the arrows in the drawing above.

Each crosspoint in our projection represents an undercrossing and an overcrossing in K . We now divide K into segments, each starting at an undercrossing and traveling in the assigned direction to the next undercrossing. By assigning distinct labels to each segment we have a set which generates the knot group.

Here is a mild justification of this. Each segment represents a distinct loop in $\mathbb{R}^3 - K$. The loop starts at some arbitrary point, Q , and travels around only that segment (right-hand justified to the

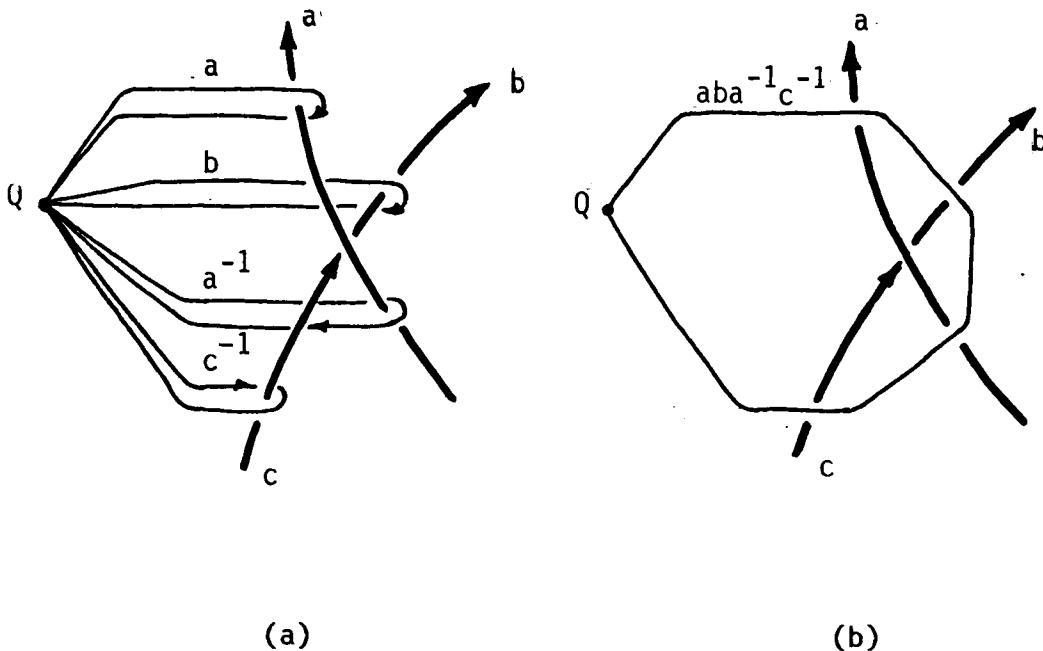
direction of that segment) and back to Q. Such loops for the trefoil are labeled a, b, and c below.



Any path in $\mathbb{R}^3$ -K can be associated with a word in the knot group by consecutively concatenating the labels of the segments which the loop passes under or over (inverses appropriately applied when the path passes a segment against the right-hand orientation). This word represents a sequence of simple loops which, when traveled consecutively, form a path equivalent to our original path.

Relators may be found by examining the crosspoints. Consider traveling a clockwise path around a crosspoint. You should cross over knot segments exactly four times. Equate passing over knot segment, s , as it leaves the crosspoint with the generator, s . Similarly, equate passing over knot segment, s , as it enters the crosspoint with s^{-1} . Then our trip around the crosspoint represents a product of four generators. This is an element of our knot group. The elements for the knot group equated with trips around all but any one crosspoint form a set of relators for our knot group.

Here again is a mild justification. Consider from diagram (a) below the path formed by traveling path a , then path b , then a^{-1} and c^{-1} . This path is equivalent to the path in diagram (b) which is the identity path. Now $aba^{-1}c^{-1}$ is the knot group element our method would extract and $aba^{-1}c^{-1} = 1$. Therefore, $aba^{-1}c^{-1}$ is a relator. The fact that this is a sufficient collection of relators requires a substantial proof not to be included here.*



With a set of generators and a collection of relators, we now have a presentation and our knot group is well-defined. Difficulties

*Crowell, Richard H., and Ralph H. Fox, Introduction to Knot Theory (New York: Springer-Verlag, 1963), p. 83.

in comparing knot groups force us to look further to less powerful invariants. By weaker invariants we mean that they may not distinguish between some knots which the knot group could show to be nonequivalent. These losses occur seldom enough that the benefits of easier comparison merit the further exploration.

Further invariants will be evaluated via the knot group. At this point we could simplify the presentation with Tietze transformations. This could ease further computations if the algorithm were to be executed entirely by hand. The rest of the algorithm, however, is to be executed by a microcomputer. The subject of algorithmetizing Tietze transformations is so complicated that it requires a branch of study all its own. Therefore, Tietze transformations must be left out of the program. Even if done by hand, the Tietze transformations tend to unformat the presentation making programming complicated. Since the microcomputer can survive without the simplifications offered by Tietze transformations, we choose to leave them out of the algorithm altogether.

With a presentation at hand, we are ready to create our Jacobian matrix. To do this we will borrow a collection of devices from free calculus called derivatives. These derivatives are functions from the integral ring of the free group on n generators (where n is the number of generators in our presentation) into itself. Although we will only define derivatives on the group elements, they yield elements of the integral ring. Therefore, it is important at this point to gain some understanding of the integral ring, hereinafter referred to as the group ring.

The group ring is defined as the set of all mappings, v , from the group into the integers for which $v(g) = 0$ for all but finitely many group elements, g . The group ring is best viewed as the set of all integral combinations of the free group elements. It can be shown that this is, in fact, a ring and that the group is a subset of this ring.

Now, for each generator of our presentation we shall define a unique derivative called the partial with respect to that particular generator. Given a group presentation

$$|g_1, g_2, \dots, g_n; r_1, r_2, \dots, r_m| \quad (1-1)$$

and an arbitrary group element of the free group (where $\epsilon_i = \pm 1$),

$$w = g_{i_1}^{\epsilon_1} g_{i_2}^{\epsilon_2} \dots g_{i_n}^{\epsilon_n},$$

we define the partial derivative of w with respect to g_i by

$$\frac{\partial w}{\partial g_j} = \sum_{k=1}^n \delta_{j, i_k} \epsilon_k \left(\prod_{\ell=1}^{k-1} g_{i_\ell}^{\epsilon_\ell} \right) g_{i_k}^{\frac{1}{2}(\epsilon_k - 1)}$$

where $\prod_{\ell=1}^0 g_{i_\ell}^{\epsilon_\ell}$ is understood to be one. For example,

$$\frac{\partial a^{-1}bab^{-1}}{\partial a} = -a^{-1} + a^{-1}b$$

and

$$\frac{\partial a^{-1}bab^{-1}}{\partial b} = a^{-1} - a^{-1}bab^{-1}.$$

The Jacobian matrix is defined in terms of these derivatives applied to the relators of our knot group. Given the group presentation in Figure 1-6, the Jacobian matrix is defined

$$M = \begin{bmatrix} \frac{\partial r_1}{\partial g_1} & \dots & \frac{\partial r_1}{\partial g_n} \\ \dots & \dots & \dots \\ \frac{\partial r_m}{\partial g_1} & \dots & \frac{\partial r_m}{\partial g_n} \end{bmatrix} \quad (1-2)$$

The Alexander polynomial is simply the reduction to a single variable of the determinant of the Jacobian matrix. Thus far we have presented a good idea of the origin of the Alexander polynomial. In practice, however, a more streamlined algorithm will be used to ease computation.

We are guaranteed by the structure of knot group presentations to have one fewer relators than generators. This means the matrix, in (1-2) is not square. Since all square matrices obtained by the elimination of any single column yield equivalent polynomials, we ignore the last generator in the presentation.

Creation of the transpose of the Jacobian matrix is considerably easier for the computer. The transposition of the Jacobian matrix has no significant effect on the determinant. Incorporating these two simplifications, we redefine the matrix

$$M = \begin{bmatrix} \frac{\partial r_1(t)}{\partial g_1} & \dots & \frac{\partial r_m(t)}{\partial g_1} \\ \dots & \dots & \dots \\ \frac{\partial r_1(t)}{\partial g_{n-1}} & \dots & \frac{\partial r_m(t)}{\partial g_{n-1}} \end{bmatrix} \quad (1-3)$$

The matrix in (1-3) is now square since $m = n - 1$.

Reduction to a single variable can be performed on the matrix elements before the determinant is evaluated. This also has no significant effect on the polynomial. Since this eases computation of both the matrix and the determinant, we incorporate this reduction into our derivative. The new derivative is defined by

$$\frac{\partial w}{\partial g_j}(t) = \sum_{k=1}^n \delta_{j,i_k} \epsilon_k \cdot t^{\sum_{\ell=1}^{k-1} \epsilon_\ell} \cdot t^{\frac{1}{2}(\epsilon_k-1)}$$

where $\sum_{\ell=1}^0 \epsilon_\ell$ is assumed to be zero.

Now we have a sleek algorithm for evaluating knots and a valuable tool for the classification of knots. The implementation of this algorithm on a microcomputer is the topic of the next chapter.

Only a few simple rules are needed to interpret knot polynomials. First, knot polynomials are said to be equivalent if they differ only by a factor of $\pm t^n$ for any integer, n . Second, knots with equivalent polynomials are not necessarily equivalent knots. However, knots which possess non-equivalent polynomials are truly non-equivalent. Lastly, any knot with a polynomial equivalent to one is a trivial knot.*

Several examples are given in appendix A. Each example includes a drawing of a projection of the knot and at least one Alexander polynomial of the knot. Some of the knot polynomials were found in literature but many were solved by hand. Each was solved again by the computer.

Notice in example 2 that different presentations for that knot yield polynomials which are equivalent by our first rule above. Notice also that examples 5 and 6 have equivalent polynomials. Although it is a substantial task, it can be shown that they are non-equivalent knots. Finally, two polynomials have been calculated in example 1. Each is equivalent to one. Hence, the knot is equivalent to the trivial knot. This fact, however, is obvious when the projection of the knot is examined.

*Crowell and Fox, p. 162.

CHAPTER II

THE COMPUTER PROGRAM

Evaluation of a knot polynomial is of frequent importance to a knot theorist. The rapid execution of such tedious tasks could speed research considerably. This program removes the responsibility of calculating Alexander polynomials from the shoulders of knot theorists. In so doing, it exemplifies that abstract tasks can be delegated to even some of the smallest of computers.

Although many mathematicians have available large mainframe computers, many do not and many would prefer to avoid the cost and trouble of large machines. One of the most important features of this program is that it handles quickly the manipulation of moderately sized data areas within the restrictions of a microcomputer.

Massive memories and high-speed processors are luxuries not available in the microcomputer world. Therefore, topics concerning choice of processor and language, and efficient software design, become important to this thesis.

The automation of any process is never without disadvantages and costs. The chief problem here is the cost of the equipment required to handle this program. The fact that computer equipment is rapidly becoming more affordable makes this program only more valuable. Careful consideration of memory and the minimal processor power needed has allowed this program to fit into a microcomputer.

The particular machine involved in this project, the TRS-80* model I, is one of the least expensive and most prolific of the microcomputer spectrum. Although a disk system and forty-eight kilobytes of memory were used to develop the program, it executes in a non-disk, sixteen kilobyte system costing half as much.

Another consideration in the automation of a task concerns what specific cases the algorithm will not handle. This program has two main restrictions. First, the relators must be exactly four generators in length. This is the format of raw relators just extracted from the knot; however, there may be some instances in which such relators are not available. In such cases the problem may be solved by Tietze transformations. Such transformations may lead to the introduction of new generators. This would be acceptable if it were not for our second restriction.

The program can handle at most nine generators. Changing the number of generators by means of Tietze transformations is almost guaranteed to unformat the relators and we find ourselves chasing our tail. The only genuine solution to the problem is to change the program code. This is a task which is easy only for an experienced Z-80 programmer and increasing the number of generators to be handled has its limits. The data areas get large very fast as the number of generators increases. The topics of data volume and such limits are discussed later.

*TRS-80 is a registered trademark of the Tandy Corporation.

The program is written in Z-80 assembly language. Because BASIC has many advantages, it was originally conceived that the program be written in BASIC, at least in part. It bears pointing out how BASIC's disadvantages outweigh its advantages and why assembler was chosen instead.

Programming in BASIC is fast and easy. BASIC is an easy language to document and understand. Most important, almost all computers have some form of BASIC, particularly the machine I was using. However, Interpreted BASIC, which is the most common among microcomputers, is relatively slow. There are vague inconsistencies in the BASIC languages used by different computers. These inconsistencies can cause great trouble in transferring code to another machine. The biggest problem with BASIC, though, is its inefficiency with memory. The algorithm involves large data structures packed with quite small values. Storing many of these values in anything more than a single byte would be a tremendous waste of precious memory. BASIC assigns no less than five bytes to any single variable. Using BASIC, the allocation of storage for the largest possible matrix immediately used up the entire sixteen kilobytes that I wanted the package to fit into. This left no room for other data or the program itself. Therefore, I chose to write the program in the more flexible assembly language.

Assembly language is hard to write, debug, alter, and almost impossible for the nonprogrammer to understand. These are disadvantages of assembler that are hard to answer for but they have little to do with the application of the program. Assembler is also heavily

machine dependent. It would be almost impossible to translate the program to a machine with a different processor without going back to the flowchart and essentially starting from scratch. Nevertheless, there are advantages which far outweigh these disadvantages.

The Z-80 microprocessor is quite popular. Moving the program to a different Z-80 based machine would require no more than perhaps some simple relocation in memory and altering the input/output routines. The input/output sections of this program were intentionally written as separate subroutines, apart from the main program, to make such alterations easy. All memory location references are by label in the mnemonics so that relocation is easy also.

Z-80 assembler has many characteristics which make it near to ideal for this application. First, assembly language is very fast. Assembler programs can run up to three hundred times faster than their BASIC counterparts. This particular program requires significantly many iterated data manipulations when applied to large problems. The speed of assembler is therefore important.

Second, assembler executable code is very compact in memory. BASIC programs could occupy as much as six times the memory as equivalent assembler code. With the tremendous data areas involved in this program, memory cannot be wasted. Hence, this feature is important.

As mentioned before, assembly language programming allows the programmer to create his own data structures. This is a necessity since the data associated with knots is rather bizarre and its volume must be minimized.

The Z-80 microprocessor has a large and powerful instruction set. As a processor with a long line of powerful ancestors, it fares well against its eight bit peers in computing power. Many features of its instruction set are well suited for this application, particularly the stack operations and the indexing registers. The latter was most helpful in creating the recursive determinant routine. In general, Z-80 assembly language became a necessity rather than an alternative to BASIC. The ultimate power of the computer necessary to this task can be realized only through assembly language.

Let us begin discussing the program itself with an overview of the nontransparent features. First the program prints an introductory message and prompts the user for a single keystroke indicating the number of generators to be used. Erroneous entries are ignored.

Next, the program displays a list of labels to be assigned to the generators and prompts for relators. A special input feature is included so that inverse generators can be entered by using the shift key. They are echoed with a trailing apostrophe. After entering four generators the program automatically invokes a carriage return so that no more than four generators can be entered per relator. Also included is a backspace feature which allows the correction of any error within that particular relator for all except the last character. The program ignores any detectable erroneous keystrokes such as characters not within the label list.

Upon entry of the last character of the last relator, the relators are redisplayed for examination and the user is given the

opportunity to reenter the entire relator set or continue. Interactivity of the user ends at the point at which the continue option is chosen.

Computation should occur rapidly and the Alexander polynomial is displayed. Terms are expressed in the variable T and exponents are indicated by a leading up arrow. At any keystroke the program restarts itself.

It was pointed out earlier in this chapter that careful selection of language and processor is important to the efficiency of a computer project. An even more important consideration, however, is software design. This section points out details of that design that were most important and gives some explanation to parts of the listing and flowchart.

The first block is labeled system initialization. The program must handle some housekeeping before any computation or input/output can begin. These housekeeping tasks include disabling the computer from interrupting program execution, assigning the memory area for stacked variables, clearing the screen and turning on the cursor. These are only mentioned because they appear in the flowchart and listing.

Memory considerations dictate the maximum number of generators the program can handle. The fact that a single keystroke determines the number of generators to be used streamlined this block considerably. It should be noted that attempts to allow the entry of larger numbers at this point will require considerably more code to maintain the same error checks.

The relator input block has two important features. First, the automatic carriage return on the entry of the fourth generator guarantees that the operator can enter no more than four generators. This avoids errors that could be hard to detect beyond this point. Second, the shifted inverse feature not only makes data entry sleek but it also serves to reduce memory required and later computations.

Although inverse generators are displayed as the generator followed by an apostrophe, the input routine actually returns to the data area the ASCII coded data byte with bit seven set. This allows inverted generators to be stored in a single byte just as noninverted generators are, and eliminates the need for translation later.

Both of these features leave the relators formatted in contiguous groups of four contiguous bytes each. This eliminates the need for memory consuming delimiters which have to be eliminated by later computation.

The method of editing the relator list is no more than adequate. Adding editing features might increase ease of using the program but only at the cost of more bulky code.

Once the relators are entered and verified by the user, the differentiation routine creates the Jacobian matrix. The last generator need not be used since any square matrix will suffice. The transpose of the Jacobian matrix is actually what is created and used. This allows the program to scan all the relators while taking partial derivatives with respect to a single generator. The resulting matrix is equivalent and this method is quicker than the reverse.

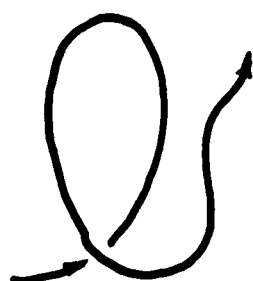
The determinant is calculated by a recursive subroutine. It returns a 1×1 matrix immediately but calls itself to find determinants of appropriate submatrices. This determinant subroutine makes up 15% of the total code and is responsible for the majority of the total execution time.

Like terms are combined as the determinant is being calculated to reduce the memory necessary to hold the various polynomials. When the determinant has been taken, each term is then displayed between keystrokes. This allows the user to control the scroll. Another keystroke runs the program again.

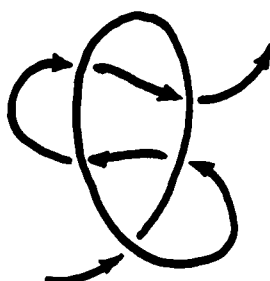
Incorrect assignment of memory to data areas could either waste valuable memory or cause overlapping of data areas. The latter would yield unreliable output with no indication to the user. Hence, data organization and the size of the data areas require careful calculation. Our data areas are largest when nine generators are used. Hence, we should calculate all data areas assuming that case.

Before we continue, an important simplification must be discussed. Segments of the knot, which we will call simple trivialities, are discribed graphically by the drawings on the following page. They are frequently obvious and easy to remove before the presentation is extracted. Even if they cannot be seen, they generate a characteristic relator with more than two appearances of the same generator. Although removing all of the simple trivialities is not always necessary, it does reduce the number of generators. More importantly, data volumes are calculated assuming that no

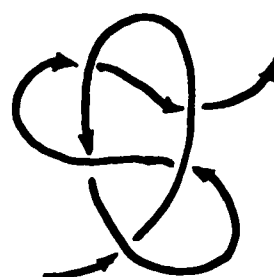
simple trivialities are present. This only becomes crucial when the limiting number of generators is used. Hence, to guarantee reliability in such cases all simple trivialities should be removed. This can be accomplished by simply straightening the portion of the knot beginning and ending with the crosspoint from which the three-generator relator was extracted.



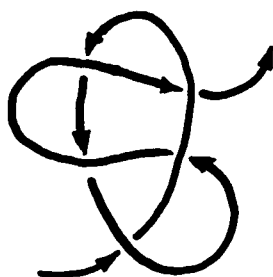
SIMPLE
TRIVIALITY



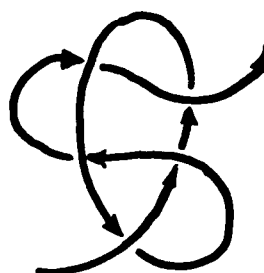
SIMPLE
TRIVIALITY



NON-
TRIVIALITY



NON-SIMPLE
TRIVIALITY



NON-SIMPLE
TRIVIALITY

Some numerical values in the program are stored as single bytes. This means that all such values must be integers between minus one hundred twenty-eight and one hundred twenty-seven. All

terms of the matrix elements have coefficients of one, minus one or zero and no exponent can exceed two in absolute value. Hence, matrix data fits easily into the byte constraints regardless of the number of generators used.

The determinant of a matrix is best viewed as the sum of signed elementary products of matrix elements. These matrix elements are chosen in such a way that no product involves any two entries from the same column or the same row. When nine generators are used, our Jacobian matrix is 8×8 . By using information about the matrix entries given in the paragraph above, we know that no exponent in any product can exceed sixteen in absolute value. Summing these products has no effect on the bounds of the exponents. Therefore, exponents of the determinant fit easily into the constraints of byte storage.

To establish an upper bound on the determinant coefficients, we must assume that no simple trivialities are present. If so, then each column has at most one entry with two non-zero terms and at most two entries with one non-zero term each. The remaining entries are all zero. Since they yield zero when included in a product, such entries are ignored.

If a particular product has i two-term entries, then the coefficient is bounded by $\pm [\sum_{j=0,i}^{\max} \binom{i}{j}]$. Since there are $\binom{8}{i}$ ways to choose i two terms entries and 2^{8-i} ways to choose the remaining one-term entries, then there are $\binom{8}{i} \times 2^{8-i}$ such products. Summing all such products and for all possible values of i , we know our determinant has coefficients which do not exceed

$$\sum_{i=0}^8 \binom{8}{i} 2^{8-i} \max_{j=0,1} \binom{i}{j} = 24310$$

in absolute value. Coefficients are stored in two byte words which can handle values of -32768 to 32767. Hence, we should feel no threat of overflow.

Now each matrix element requires eight bytes (four terms of one coefficient and one exponent each). When nine generators are used as the determinant is computed, memory may have to contain simultaneously eight different square matrices of sizes 1 x 1 to 8 x 8. Therefore, we can predict the matrix area to require

$$8 \sum_{i=1}^8 i^2 = 1632 \text{ bytes.}$$

The number of terms in the determinant depends upon the character of the Jacobian matrix. We do know, however, that no exponent in our determinant can exceed sixteen in absolute value. Hence, after like terms are combined, the determinant will have at most thirty-three terms. Each term requires three bytes (two for the coefficient and one for the exponent) so our determinant will fit into an area of ninety-nine bytes.

To be computed, however, one such polynomial area must be reserved for the determinant of our Jacobian matrix and each submatrix created by the determinant routine. An additional seventeen

bytes must be allowed for larger polynomials present before like terms are combined and for delimiters. Hence, the memory required to handle the determinant as it is being computed is

$$15 + \sum_{i=1}^8 12i + 3 = 473 \text{ bytes.}$$

As products of polynomials are being computed, a buffer is reserved to store them. Before like terms are combined this buffer could have to store up to sixty-six terms. Room for these terms and delimiters comes to an additional two hundred bytes.

The program and data areas come to a total of 4,180 bytes. This fits into the standard sixteen kilobytes offered on most micro-computers.

LIST OF REFERENCES

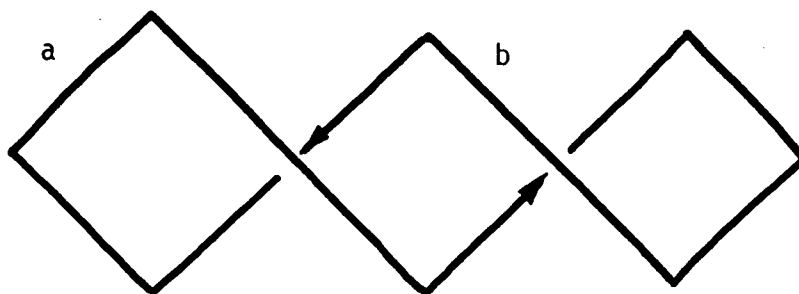
LIST OF REFERENCES

- Barden, William, Jr., TRS-80 Assembly-Language Programming; Tandy Corporation Company, Fort Worth, Texas, 1979.
- Crowell, Richard H., and Ralph H. Fox, Introduction to Knot Theory; Springer-Verlag, New York, New York, 1963.
- Fox, R.H., "A Quick Trip Through Knot Theory"; Topology of 3-Manifolds and Related Topics, M.K. Fort, Jr., Editor; Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1962.
- Merrin, Stephen, "Addition and Subtraction: The 1802 Versus the Z80", BYTE The Small Systems Journal; McGraw-Hill, Vol. 6, No. 3, March, 1981.

APPENDICES

APPENDIX A
EXAMPLES

EXAMPLE 1



$$\Pi_1(\mathbb{R}^3 - K) = |a, b : aa^{-1}b^{-1}a|$$

$$M = [1 - 1 + t^{-1}]$$

Alexander Polynomial
(computed and by hand)

$$P(t) = t^{-1}$$

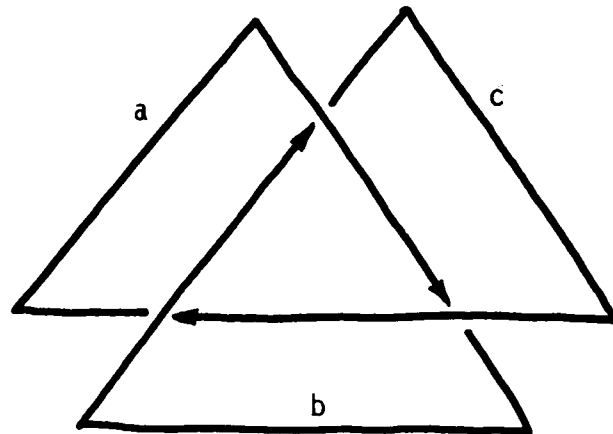
$$\Pi_1(\mathbb{R}^3 - K) = |a, b : bb^{-1}ab|$$

$$M = [-t^{-1}]$$

Alexander Polynomial
(computed and by hand)

$$P(t) = -t^{-1}$$

EXAMPLE 2



$$\Pi_1(\mathbb{R}^3 - K) = \langle a, b, c : abc^{-1}b^{-1}, cab^{-1}a^{-1} \rangle$$

$$M = \begin{bmatrix} 1 & t-1 \\ t-1 & -t \end{bmatrix}$$

Alexander Polynomial
(computed and by hand)

$$P(t) = -1 + t - t^2$$

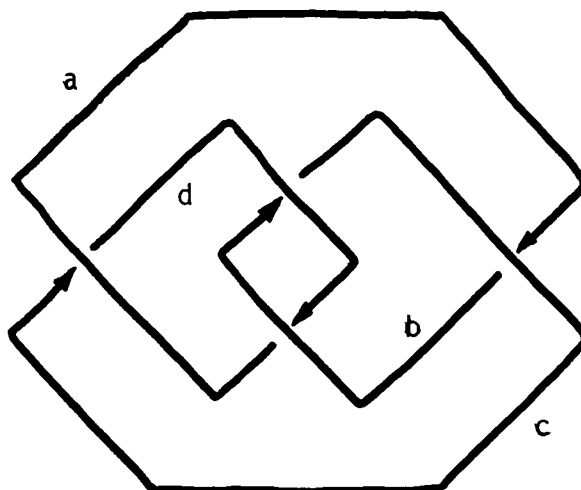
$$\Pi_1(\mathbb{R}^3 - K) = \langle a, b, c : cab^{-1}a^{-1}, bca^{-1}c^{-1} \rangle$$

$$M = \begin{bmatrix} t-1 & -t \\ -t & 1 \end{bmatrix}$$

Alexander Polynomial
(computed and by hand)

$$P(t) = -1 + t - t^2$$

EXAMPLE 3



$$\Pi_1(\mathbb{R}^3 - K) = \langle a, b, c, d : da^{-1}c^{-1}a, bc^{-1}a^{-1}c, abd^{-1}b^{-1} \rangle$$

$$M = \begin{bmatrix} -1 + t^{-1} & -t^{-1} & 1 \\ 0 & 1 & t-1 \\ -t^{-1} & -1+t^{-1} & 0 \end{bmatrix}$$

Alexander Polynomial
(computed and by hand)

$$P(t) = -t^{-1} + 3 - t$$

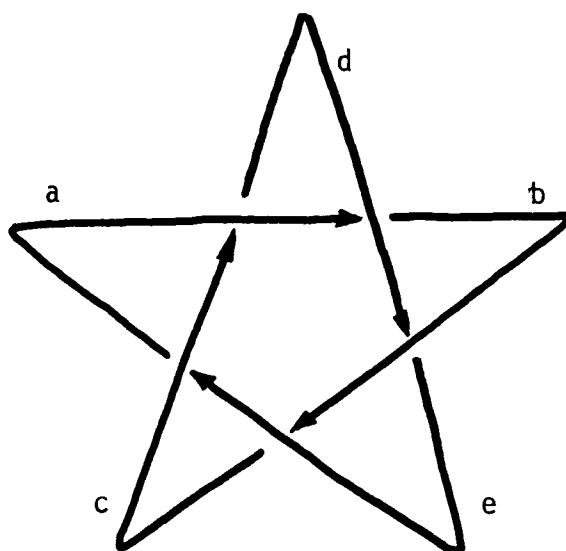
$$\Pi_1(\mathbb{R}^3 - K) = \langle a, b, c, d : bc^{-1}a^{-1}c, abd^{-1}b^{-1}, cdb^{-1}d^{-1} \rangle$$

$$M = \begin{bmatrix} -t^{-1} & 1 & 0 \\ 1 & t-1 & -t \\ -1+t^{-1} & 0 & 1 \end{bmatrix}$$

Alexander Polynomial
(computed and by hand)

$$P(t) = t^{-1} - 3 + t$$

EXAMPLE 4



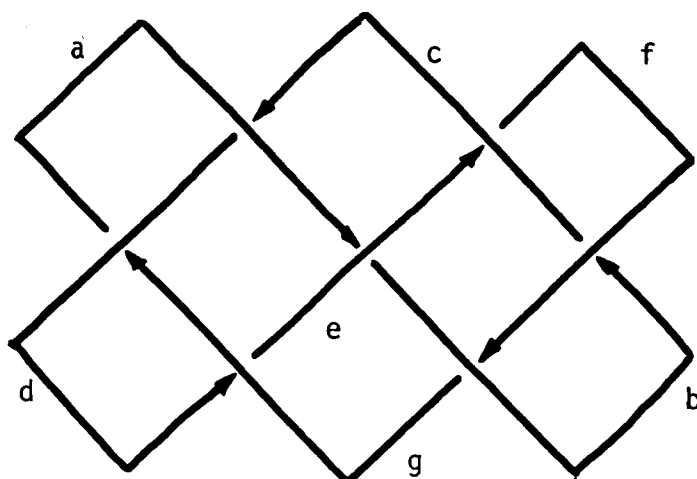
$$\Pi_1(\mathbb{R}^3 - K) = |a, b, c, d, e : ace^{-1}c^{-1}, bda^{-1}d^{-1}, ceb^{-1}e^{-1}, dac^{-1}a^{-1}|$$

$$M = \begin{bmatrix} 1 & -t^2 & 0 & t-1 \\ 0 & 1 & -t^2 & 0 \\ t-1 & 0 & 1 & -t^2 \\ 0 & t-1 & 0 & 1 \end{bmatrix}$$

Alexander Polynomial
(computed and by hand)

$$P(t) = 1 - t + t^2 - t^3 + t^4$$

EXAMPLE 5



$$\Pi_1(\mathbb{R}^3 - K) = \langle a, b, c, d, e, f, g : ad^{-1}g^{-1}d, be^{-1}a^{-1}e, cf^{-1}b^{-1}f, da^{-1}c^{-1}a, \\ eg^{-1}d^{-1}g, fc^{-1}e^{-1}c \rangle$$

Alexander Polynomial*
(from literature)

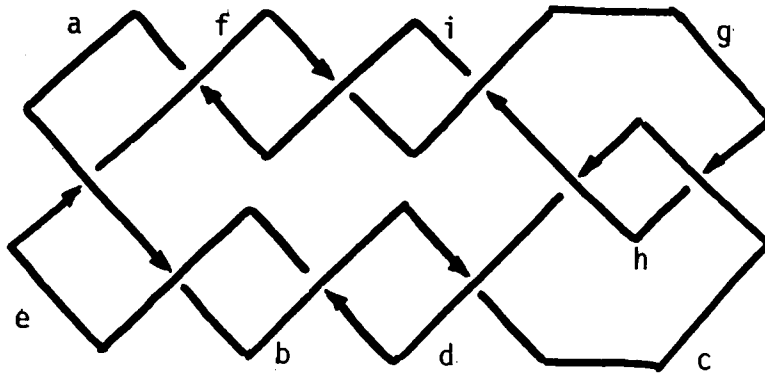
$$P(t) = 4t^2 - 7t + 4$$

Alexander Polynomial
(computed)

$$P(t) = 4t^{-4} - 7t^{-3} + 4t^{-2}$$

*Crowell and Fox, p. 130.

EXAMPLE 6



$$\Pi_1(\mathbb{R}^3 - K) = \langle a, b, c, d, e, f, g, h, i : a f i^{-1} f^{-1}, b e a^{-1} e^{-1}, d h e^{-1} h^{-1}, e b d^{-1} b^{-1}, \\ f a e^{-1} a^{-1}, g i f^{-1} i^{-1}, h c g^{-1} c^{-1} \rangle$$

Alexander Polynomial*
(from literature)

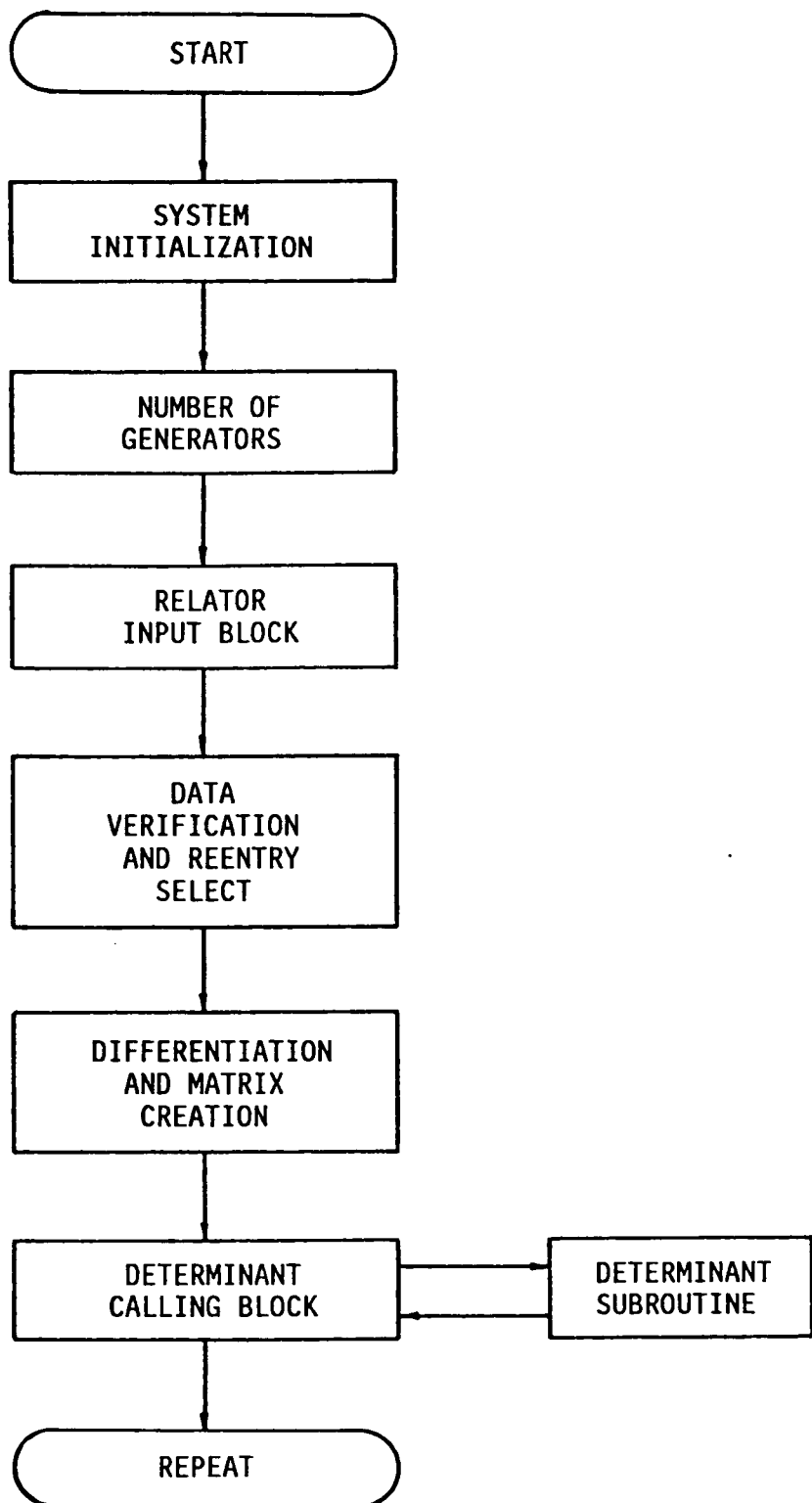
$$P(t) = 4t^2 - 7t + 4$$

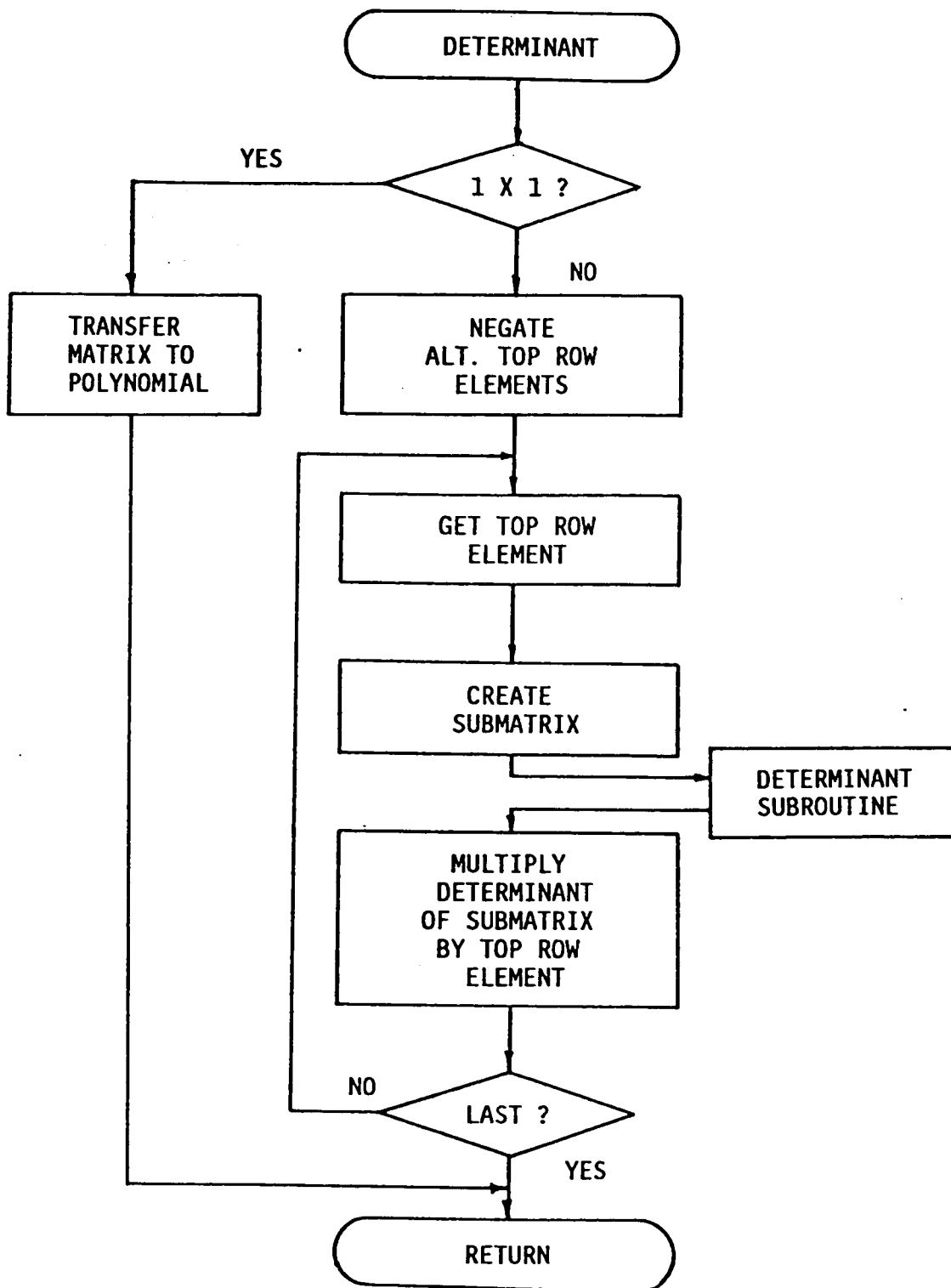
Alexander Polynomial
(computed)

$$P(t) = 4t^3 - 7t^4 + 4t^5$$

*Crowell and Fox, p. 130.

APPENDIX B
FLOWCHARTS





APPENDIX C

LISTING

	00100 ;		
	00110 ;		COMPUTER EVALUATION
	00120 ;		OF TOPOLOGICAL KNOTS
	00130 ;		
6000	00140	ORG	6000H
	00150 ;		
	00160 ;		SYSTEM INITIALIZATION
	00170 ;		
6000 31FA7F	00180 THESIS	LD	SP,7FFAH
6003 CD0664	00190	CALL	CLS
6006 3E0E	00200	LD	A,0EH
6008 CD1D64	00210	CALL	DISP
600B 213465	00220	LD	HL,AMSG
600E CDF463	00230	CALL	PRINT
	00240 ;		
	00250 ;		NUMBER OF GENERATORS
	00260 ;		INPUT BLOCK
	00270 ;		
6011 CDE964	00280 REDO	CALL	INKEY
6014 326B65	00290	LD	(DOTA),A
6017 C610	00300	ADD	A,10H
6019 328B65	00310	LD	(DOTB),A
601C DD218B65	00320	LD	IX,DOTB
6020 D641	00330	SUB	41H
6022 38ED	00340	JR	C,REDO
6024 28EB	00350	JR	Z,REDO
6026 FE09	00360	CP	09H
6028 30E7	00370	JR	NC,REDO
602A 323365	00380	LD	(NUM),A
	00390 ;		
	00400 ;		RELATOR INPUT BLOCK
	00410 ;		
602D F5	00420	PUSH	AF
602E CD0664	00430	CALL	CLS
6031 214A65	00440	LD	HL,BMSG
6034 CDF463	00450	CALL	PRINT
6037 DD3400	00460	INC	(IX)
603A C1	00470	POP	BC
603B 218C66	00480	LD	HL,RELS
603E 0E04	00490	LD	C,4
6040 C5	00500 AGAIN	PUSH	BC
6041 E5	00510	PUSH	HL
6042 CDE964	00520	CALL	INKEY
6045 21C03F	00530	LD	HL,3FC0H
6048 0618	00540	LD	B,18H
604A 3620	00550 CMES	LD	(HL),20H
604C 23	00560	INC	HL
604D 10FB	00570	DJNZ	CMES
604F E1	00580	POP	HL
6050 C1	00590	POP	BC
6051 FE08	00600	CP	08H
6053 2014	00610	JR	NZ,NBCKSP

6055 5F	00620	LD	E,A
6056 79	00630	LD	A,C
6057 FE04	00640	CP	4
6059 2BE5	00650	JR	Z, AGAIN
605B 0C	00660	INC	C
605C 2B	00670	DEC	HL
605D 7B	00680	LD	A,E
605E CB7E	00690	BIT	7, (HL)
6060 C41D64	00700	CALL	NZ, DISP
6063 7B	00710	LD	A,E
6064 CD1D64	00720	CALL	DISP
6067 1BD7	00730	JR	AGAIN
6069 5F	00740 NBCKSP	LD	E,A
606A CBBF	00750	RES	7,A
606C FE41	00760	CP	41H
606E 3806	00770	JR	C, INPERR
6070 DDBE00	00780	CP	(IX)
6073 7B	00790	LD	A,E
6074 3811	00800	JR	C, LOAD
6076 E5	00810 INPERR	PUSH	HL
6077 217466	00820	LD	HL, ZMSG
607A C5	00830	PUSH	BC
607B 011600	00840	LD	BC, 0016H
607E 11C03F	00850	LD	DE, 3FC0H
6081 EDB0	00860	LDIR	
6083 C1	00870	POP	BC
6084 E1	00880	POP	HL
6085 18B9	00890	JR	AGAIN
6087 77	00900 LOAD	LD	(HL), A
6088 23	00910	INC	HL
6089 CD1D64	00920	CALL	DISP
608C 0D	00930	DEC	C
608D 20B1	00940	JR	NZ, AGAIN
608F 3E0D	00950	LD	A, 0DH
6091 CD1D64	00960	CALL	DISP
6094 0E04	00970	LD	C, 4
6096 10A8	00980	DJNZ	AGAIN
	00990 ;		
	01000 ;		
	01010 ;		
	01020 ;		
6098 CD0664	01030	CALL	CLS
609B 21D365	01040	LD	HL, CMSG
609E CDF463	01050	CALL	PRINT
60A1 218C66	01060	LD	HL, RELS
60A4 3A3365	01070	LD	A, (NUM)
60A7 4F	01080	LD	C, A
60AB 0604	01090 LINELP	LD	B, 4
60AA 7E	01100 CHARLP	LD	A, (HL)
60AB CD1D64	01110	CALL	DISP
60AE 23	01120	INC	HL
60AF 10F9	01130	DJNZ	CHARLP

DATA VERIFICATION AND
REENTRY SELECT

60B1	3E0D	01140	LD	A,ODH
60B3	CD1D64	01150	CALL	DISP
60B6	0D	01160	DEC	C
60B7	20EF	01170	JR	NZ,LINELP
60B9	CDE964	01180	CALL	INKEY
60BC	FE32	01190	CP	'R'
60BE	CA0060	01200	JP	Z,THESIS
60C1	FE43	01210	CP	'C'
60C3	20F4	01220	JR	NZ,ANOTHR
		01230 ;		
		01240 ;		
		01250 ;		
		01260 ;		
			DIFFERENTIATION AND	
			MATRIX CREATION	
60C5	21F469	01270	LD	HL,MATRIX
60C8	3A3365	01280	LD	A,(NUM)
60CB	47	01290	LD	B,A
60CC	0E40	01300	LD	C,40H
60CE	0C	01310	INC	C
60CF	C5	01320	PUSH	BC
60D0	3A3365	01330	LD	A,(NUM)
60D3	47	01340	LD	B,A
60D4	DD218C66	01350	LD	IX,RELS
60D8	C5	01360	PUSH	BC
60D9	0604	01370	LD	B,4
60DB	1600	01380	LD	D,0
60DD	DDCB007E	01390	BIT	7,(IX)
60E1	2006	01400	JR	NZ,INVRTD
60E3	5A	01410	LD	E,D
60E4	14	01420	INC	D
60E5	3E01	01430	LD	A,1
60E7	1804	01440	JR	SKIP
60E9	15	01450	DEC	D
60EA	5A	01460	LD	E,D
60EB	AF	01470	XOR	A
60EC	3D	01480	DEC	A
60ED	F5	01490	PUSH	AF
60EE	DD7E00	01500	LD	A,(IX)
60F1	CBBF	01510	RES	7,A
60F3	B9	01520	CP	C
60F4	2B06	01530	JR	Z,MATCH
60F6	F1	01540	POP	AF
60F7	AF	01550	XOR	A
60F8	1E01	01560	LD	E,1
60FA	1801	01570	JR	LOADM
60FC	F1	01580	POP	AF
60FD	77	01590	LD	(HL),A
60FE	23	01600	INC	HL
60FF	7B	01610	LD	A,E
6100	77	01620	LD	(HL),A
6101	23	01630	INC	HL
6102	DD23	01640	INC	IX
6104	10D7	01650	DJNZ	TERMLP

6106 C1	01660		POP	BC
6107 10CF	01670		DJNZ	COLLP
6109 C1	01680		POP	BC
610A 10C2	01690		DJNZ	ROWLP
	01700 ;			
	01710 ;			
	01720 ;			
	01730 ;			
			DETERMINANT	
			CALLING BLOCK	
610C DD218C66	01740		LD	IX,RELS
6110 DD7503	01750		LD	(IX+3),L
6113 DD7404	01760		LD	(IX+4),H
6116 21F469	01770		LD	HL,MATRIX
6119 DD7501	01780		LD	(IX+1),L
611C DD7402	01790		LD	(IX+2),H
611F 3A3365	01800		LD	A,(NUM)
6122 DD7700	01810		LD	(IX),A
6125 215367	01820		LD	HL,POLY
6128 DD7505	01830		LD	(IX+5),L
612B DD7406	01840		LD	(IX+6),H
612E DD7507	01850		LD	(IX+7),L
6131 DD7408	01860		LD	(IX+8),H
6134 CD8461	01870		CALL	DETER
6137 CD0664	01880		CALL	CLS
613A 212166	01890		LD	HL,JMSG
613D CDF463	01900		CALL	PRINT
6140 FD215367	01910		LD	IY,POLY
6144 FD6600	01920	SHLOOP	LD	H,(IY)
6147 FD23	01930		INC	IY
6149 FD6E00	01940		LD	L,(IY)
614C 7C	01950		LD	A,H
614D B5	01960		OR	L
614E 2B2B	01970		JR	Z,FINSH
6150 CD9D63	01980		CALL	DWD
6153 215266	01990		LD	HL,LMSG
6156 CDF463	02000		CALL	PRINT
6159 FD23	02010		INC	IY
615B FD6E00	02020		LD	L,(IY)
615E FD23	02030		INC	IY
6160 2600	02040		LD	H,0
6162 CB7D	02050		BIT	7,L
6164 2B02	02060		JR	Z,UNNEG
6166 26FF	02070		LD	H,OFFH
6168 3E01	02080	UNNEG	LD	A,1
616A CD9D63	02090		CALL	DWD
616D 213966	02100		LD	HL,KMSG
6170 CDF463	02110		CALL	PRINT
6173 CDE964	02120		CALL	INKEY
6176 18CC	02130		JR	SHLOOP
6178 215766	02140	FINSH	LD	HL,MMSG
617B CDF463	02150		CALL	PRINT
617E CDE964	02160		CALL	INKEY
6181 C30060	02170		JP	THESIS

	02190 ;		DETERMINANT SUBROUTINE
	02200 ;		MAIN UNIT
	02210 ;		
6184 3E01	02220 DETER	LD	A, 1
6186 DDBE00	02230	CP	(IX)
6189 2035	02240	JR	NZ, COMPLX
	02250 ;		
	02260 ;		SIMPLE SUBUNIT
	02270 ;		
618B DD6E05	02280	LD	L, (IX+5)
618E DD6606	02290	LD	H, (IX+6)
6191 E5	02300	PUSH	HL
6192 FDE1	02310	POP	IY
6194 DD5E01	02320	LD	E, (IX+1)
6197 DD5602	02330	LD	D, (IX+2)
619A 0604	02340	LD	B, 4
619C 0E00	02350	LD	C, 0
619E 1A	02360 SIBLM	LD	A, (DE)
619F 13	02370	INC	DE
61A0 13	02380	INC	DE
61A1 B7	02390	OR	A
61A2 2B11	02400	JR	Z, ZTRM
61A4 1B	02410	DEC	DE
61A5 0E00	02420	LD	C, 0
61A7 CB7F	02430	BIT	7, A
61A9 2B02	02440	JR	Z, NNNNEG
61AB 0EFF	02450	LD	C, OFFH
61AD 71	02460 NNNNEG	LD	(HL), C
61AE 23	02470	INC	HL
61AF 77	02480	LD	(HL), A
61B0 23	02490	INC	HL
61B1 1A	02500	LD	A, (DE)
61B2 13	02510	INC	DE
61B3 77	02520	LD	(HL), A
61B4 23	02530	INC	HL
61B5 10E7	02540 ZTRM	DJNZ	SIBLM
61B7 0E00	02550	LD	C, 0
61B9 71	02560	LD	(HL), C
61BA 23	02570	INC	HL
61BB 71	02580	LD	(HL), C
61BC CD1E63	02590	CALL	CLT
61BF C9	02600	RET	
	02610 ;		
	02620 ;		COMPLEX SUBUNIT
	02630 ;		
61C0 DD6E01	02640 COMPLX	LD	L, (IX+1)
61C3 DD6602	02650	LD	H, (IX+2)
61C6 DD4600	02660	LD	B, (IX)
	02670 ;		
	02680 ;		NEGATING ALTERNATING
	02690 ;		ELEMENTS OF THE

	02700 ;	TOP ROW		
	02710 ;			
61C9 CD9563	02720 SETLP	CALL	LEAP	
61CC 05	02730	DEC	B	
61CD 280E	02740	JR	Z, SETDON	
61CF C5	02750	PUSH	BC	
61D0 0604	02760	LD	B, 4	
61D2 7E	02770 NEGLP	LD	A, (HL)	
61D3 ED44	02780	NEG		
61D5 77	02790	LD	(HL), A	
61D6 23	02800	INC	HL	
61D7 23	02810	INC	HL	
61D8 10FB	02820	DJNZ	NEGLP	
61DA C1	02830	POP	BC	
61DB 10EC	02840	DJNZ	SETLP	
	02850 ;			
	02860 ;	WORK ACROSS TOP		
	02870 ;	ROW BY ELEMENTS		
	02880 ;			
61DD DD4600	02890 SETDON	LD	B, (IX)	
61E0 DD6E01	02900	LD	L, (IX+1)	
61E3 DD6602	02910	LD	H, (IX+2)	
61E6 DD7509	02920	LD	(IX+9), L	
61E9 DD740A	02930	LD	(IX+10), H	
	02940 ;			
	02950 ;	CREATE NEW SUBMATRIX		
	02960 ;			
61EC C5	02970 TROWLP	PUSH	BC	
61ED 48	02980	LD	C, B	
61EE DD6E01	02990	LD	L, (IX+1)	
61F1 DD6602	03000	LD	H, (IX+2)	
61F4 DD4600	03010	LD	B, (IX)	
61F7 CD9563	03020 TRSKLP	CALL	LEAP	
61FA 10FB	03030	DJNZ	TRSKLP	
61FC DD5E03	03040	LD	E, (IX+3)	
61FF DD5604	03050	LD	D, (IX+4)	
6202 DD4600	03060	LD	B, (IX)	
6205 05	03070	DEC	B	
6206 DD700B	03080	LD	(IX+11), B	
6209 C5	03090 RLOOP	PUSH	BC	
620A DD4600	03100	LD	B, (IX)	
620D 78	03110 CLOOP	LD	A, B	
620E B9	03120	CP	C	
620F 2005	03130	JR	NZ, NSKIP	
6211 CD9563	03140	CALL	LEAP	
6214 1807	03150	JR	NEXTC	
6216 C5	03160 NSKIP	PUSH	BC	
6217 010800	03170	LD	BC, B	
621A EDB0	03180	LDIR		
621C C1	03190	POP	BC	
621D 10EE	03200 NEXTC	DJNZ	CLOOP	
621F C1	03210	POP	BC	

6220 10E7	03220	DJNZ	RLOOP
	03230 ;		
	03240 ;		SETUP FOR NEW
	03250 ;		DETERMINANT CALL
	03260 ;		
6222 DD750C	03270	LD	(IX+12),L
6225 DD740D	03280	LD	(IX+13),H
6228 DD730E	03290	LD	(IX+14),E
622B DD720F	03300	LD	(IX+15),D
622E DD5E07	03310	LD	E, (IX+7)
6231 DD5608	03320	LD	D, (IX+8)
6234 DD7310	03330	LD	(IX+16),E
6237 DD7211	03340	LD	(IX+17),D
623A DD7312	03350	LD	(IX+18),E
623D DD7213	03360	LD	(IX+19),D
6240 110B00	03370	LD	DE, 11
6243 DD19	03380	ADD	IX, DE
6245 CD8461	03390	CALL	DETER
6248 11F5FF	03400	LD	DE, -11
624B DD19	03410	ADD	IX, DE
	03420 ;		
	03430 ;		
	03440 ;		MULTIPLICATION OF
	03450 ;		POLYNOMIALS SUBUNIT
624D 212C69	03460	LD	HL, PLYPRD
6250 228A66	03470	LD	(PRDPTR), HL
6253 3600	03480	LD	(HL), 0
6255 23	03490	INC	HL
6256 3600	03500	LD	(HL), 0
6258 DD6E09	03510	LD	L, (IX+9)
625B DD660A	03520	LD	H, (IX+10)
625E E5	03530	PUSH	HL
625F FDE1	03540	POP	IY
6261 0604	03550	LD	B, 4
6263 C5	03560 MXLOOP	PUSH	BC
6264 FD7E00	03570	LD	A, (IY)
6267 B7	03580	OR	A
6268 286C	03590	JR	Z, ZMT
626A DD6E10	03600	LD	L, (IX+16)
626D DD6611	03610	LD	H, (IX+17)
6270 DDE5	03620	PUSH	IX
6272 56	03630 PYLOOP	LD	D, (HL)
6273 23	03640	INC	HL
6274 5E	03650	LD	E, (HL)
6275 23	03660	INC	HL
6276 7A	03670	LD	A, D
6277 B3	03680	OR	E
6278 2841	03690	JR	Z, ZPT
627A E5	03700	PUSH	HL
627B FD7E00	03710	LD	A, (IY)
627E 47	03720	LD	B, A

627F	AA	03730		XOR	D
6280	F5	03740		PUSH	AF
6281	210000	03750		LD	HL,0
6284	CB78	03760		BIT	7,B
6286	2804	03770		JR	Z,MENZ
6288	78	03780		LD	A,B
6289	ED44	03790		NEG	
628B	47	03800		LD	B,A
628C	CB7A	03810	MENZ	BIT	7,D
628E	C41663	03820		CALL	NZ,NWORD
6291	19	03830	MUL00P	ADD	HL,DE
6292	10FD	03840		DJNZ	MUL00P
6294	E5	03850		PUSH	HL
6295	D1	03860		POP	DE
6296	F1	03870		POP	AF
6297	CB7F	03880		BIT	7,A
6299	C41663	03890		CALL	NZ,NWORD
629C	E1	03900		POP	HL
629D	7E	03910		LD	A,(HL)
629E	23	03920		INC	HL
629F	FD8601	03930		ADD	A,(IY+1)
62A2	DD2ABA66	03940		LD	IX,(PRDPTR)
62A6	DD7200	03950		LD	(IX),D
62A9	DD23	03960		INC	IX
62AB	DD7300	03970		LD	(IX),E
62AE	DD23	03980		INC	IX
62B0	DD7700	03990		LD	(IX),A
62B3	DD23	04000		INC	IX
62B5	DD228A66	04010		LD	(PRDPTR),IX
62B9	18B7	04020		JR	PYLOOP
62BB	DD2ABA66	04030	ZPT	LD	IX,(PRDPTR)
62BF	DD7700	04040		LD	(IX),A
62C2	DD7701	04050		LD	(IX+1),A
62C5	FDE5	04060		PUSH	IY
62C7	FD212C69	04070		LD	IY,PLYPRD
62CB	CD1E63	04080		CALL	CLT
62CE	FD228A66	04090		LD	(PRDPTR),IY
62D2	FDE1	04100		POP	IY
62D4	DDE1	04110		POP	IX
62D6	FD23	04120	ZMT	INC	IY
62D8	FD23	04130		INC	IY
62DA	C1	04140		POP	BC
62DB	1086	04150		DJNZ	MXLOOP
62DD	FDE5	04160		PUSH	IY
62DF	E1	04170		POP	HL
62E0	DD7509	04180		LD	(IX+9),L
62E3	DD740A	04190		LD	(IX+10),H
62E6	DD5E10	04200		LD	E,(IX+16)
62E9	DD5611	04210		LD	D,(IX+17)
62EC	212C69	04220		LD	HL,PLYPRD
62EF	010300	04230	MLLOOP	LD	BC,3
62F2	7E	04240		LD	A,(HL)

62F3	23	04250	INC	HL
62F4	B6	04260	OR	(HL)
62F5	2B	04270	DEC	HL
62F6	EDB0	04280	LDIR	
62F8	B7	04290	OR	A
62F9	20F4	04300	JR	NZ,MLLOOP
62FB	DD6E05	04310	LD	L,(IX+5)
62FE	DD6606	04320	LD	H,(IX+6)
6301	E5	04330	PUSH	HL
6302	FDE1	04340	POP	IY
6304	CD1E63	04350	CALL	CLT
6307	FDE5	04360	PUSH	IY
6309	E1	04370	POP	HL
630A	DD7507	04380	LD	(IX+7),L
630D	DD740B	04390	LD	(IX+8),H
		04400 ;		
		04410 ;		
6310	C1	04420	POP	BC
6311	05	04430	DEC	B
6312	C2EC61	04440	JP	NZ,TROWLP
6315	C9	04450	RET	
		04460 ;		
		04470 ;		
		04480 ;		
		04490 ;		
			I/O AND OTHER	
			SUBROUTINES	
6316	7A	04500	NWORD	LD A,D
6317	2F	04510		CPL
6318	57	04520		LD D,A
6319	7B	04530		LD A,E
631A	2F	04540		CPL
631B	5F	04550		LD E,A
631C	13	04560		INC DE
631D	C9	04570		RET
		04580 ;		
631E	FDE5	04590	CLT	PUSH IY
6320	21EC66	04600		LD HL,BUFF
6323	3600	04610		LD (HL),0
6325	11ED66	04620		LD DE,BUFF+1
6328	014200	04630		LD BC,66
632B	EDB0	04640		LDIR
632D	FD4600	04650	PSCAN	LD B,(IY)
6330	FD7E01	04660		LD A,(IY+1)
6333	B0	04670		OR B
6334	2B30	04680		JR Z,WALA
6336	21EC66	04690		LD HL,BUFF
6339	FD7E02	04700		LD A,(IY+2)
633C	C610	04710		ADD A,16
633E	CB27	04720		SLA A
6340	1600	04730		LD D,0
6342	5F	04740		LD E,A
6343	19	04750		ADD HL,DE
6344	DDE5	04760		PUSH IX

6346	E5	04770		PUSH	HL
6347	DDE1	04780		POP	IX
6349	FD6600	04790		LD	H, (IY)
634C	FD6E01	04800		LD	L, (IY+1)
634F	DD5600	04810		LD	D, (IX)
6352	DD5E01	04820		LD	E, (IX+1)
6355	19	04830		ADD	HL, DE
6356	DD7400	04840		LD	(IX), H
6359	DD7501	04850		LD	(IX+1), L
635C	DDE1	04860		POP	IX
635E	FD23	04870		INC	IY
6360	FD23	04880		INC	IY
6362	FD23	04890		INC	IY
6364	18C7	04900		JR	PSCAN
6366	FDE1	04910	WALA	POP	IY
6368	0621	04920		LD	B, 33
636A	21EC66	04930		LD	HL, BUFF
636D	4E	04940	MSCAN	LD	C, (HL)
636E	23	04950		INC	HL
636F	7E	04960		LD	A, (HL)
6370	B1	04970		OR	C
6371	2813	04980		JR	Z, ZZTRM
6373	7E	04990		LD	A, (HL)
6374	FD7100	05000		LD	(IY), C
6377	FD23	05010		INC	IY
6379	FD7700	05020		LD	(IY), A
637C	FD23	05030		INC	IY
637E	3E11	05040		LD	A, 17
6380	90	05050		SUB	B
6381	FD7700	05060		LD	(IY), A
6384	FD23	05070		INC	IY
6386	23	05080	ZZTRM	INC	HL
6387	10E4	05090		DJNZ	MSCAN
6389	AF	05100		XOR	A
638A	FD7700	05110		LD	(IY), A
638D	FD23	05120		INC	IY
638F	FD7700	05130		LD	(IY), A
6392	FD2B	05140		DEC	IY
6394	C9	05150		RET	
		05160	;		
6395	C5	05170	LEAP	PUSH	BC
6396	0608	05180		LD	B, 8
6398	23	05190	LEAPLP	INC	HL
6399	10FD	05200		DJNZ	LEAPLP
639B	C1	05210		POP	BC
639C	C9	05220		RET	
		05230	;		
639D	E5	05240	DWD	PUSH	HL
639E	D5	05250		PUSH	DE
639F	C5	05260		PUSH	BC
63A0	F5	05270		PUSH	AF
63A1	B7	05280		OR	A

63A2	2810	05290	JR	Z,NNEG
63A4	CB7C	05300	BIT	7,H
63A6	280C	05310	JR	Z,NNEG
63A8	3E2D	05320	LD	A,'-'
63AA	CD1D64	05330	CALL	DISP
63AD	7C	05340	LD	A,H
63AE	2F	05350	CPL	
63AF	67	05360	LD	H,A
63B0	7D	05370	LD	A,L
63B1	2F	05380	CPL	
63B2	6F	05390	LD	L,A
63B3	23	05400	INC	HL
63B4	0E00	05410	LD	C,0
63B6	111027	05420	LD	DE,10000
63B9	CDD963	05430	CALL	DIVS
63BC	11E803	05440	LD	DE,1000
63BF	CDD963	05450	CALL	DIVS
63C2	116400	05460	LD	DE,100
63C5	CDD963	05470	CALL	DIVS
63C8	110A00	05480	LD	DE,10
63CB	CDD963	05490	CALL	DIVS
63CE	7D	05500	LD	A,L
63CF	0EFF	05510	LD	C,OFFH
63D1	CDE363	05520	CALL	SHOWD
63D4	F1	05530	POP	AF
63D5	C1	05540	POP	BC
63D6	D1	05550	POP	DE
63D7	E1	05560	POP	HL
63D8	C9	05570	RET	
63D9	3EFF	05580	LD	A,OFFH
63DB	3C	05590	INC	A
63DC	37	05600	SCF	
63DD	3F	05610	CCF	
63DE	ED52	05620	SBC	HL,DE
63E0	30F9	05630	JR	NC,DIVLP
63E2	19	05640	ADD	HL,DE
63E3	B9	05650	CP	C
63E4	CB	05660	RET	Z
63E5	B7	05670	OR	A
63E6	2005	05680	JR	NZ,NZERO
63E8	3E30	05690	LD	A,'0'
63EA	C31D64	05700	JP	DISP
63ED	C630	05710	ADD	A,30H
63EF	0EFF	05720	LD	C,OFFH
63F1	C31D64	05730	JP	DISP
		05740 ;		
		05750 ;		MESSAGE PRINT ROUTINE
		05760 ;		
63F4	7E	05770	LD	A,(HL)
63F5	B7	05780	OR	A
63F6	CB	05790	RET	Z
63F7	FE2A	05800	CP	'*'

63F9 C8	05810	RET	Z
63FA FE2F	05820	CP	'/'
63FC 2002	05830	JR	NZ, NCRLF
63FE 3E0D	05840	LD	A, 0DH
6400 CD1D64	05850 NCRLF	CALL	DISP
6403 23	05860	INC	HL
6404 18EE	05870	JR	PRINT
	05880 ;		
	05890 ;		
	05900 ;		
			CLEAR SCREEN ROUTINE
6406 C5	05910 CLS	PUSH	BC
6407 D5	05920	PUSH	DE
6408 E5	05930	PUSH	HL
6409 21003C	05940	LD	HL, 3C00H
640C 229B64	05950	LD	(CLOCN), HL
640F 11013C	05960	LD	DE, 3C01H
6412 010004	05970	LD	BC, 1024
6415 3620	05980	LD	(HL), 20H
6417 EDB0	05990	LDIR	
6419 E1	06000	POP	HL
641A D1	06010	POP	DE
641B C1	06020	POP	BC
641C C9	06030	RET	
	06040 ;		
	06050 ;		
	06060 ;		
	06070 ;		
			CHARACTER DISPLAY ROUTINE
641D CB7F	06080 DISP	BIT	7, A
641F 2B07	06090	JR	Z, DISPL
6421 CBBF	06100	RES	7, A
6423 CD2B64	06110	CALL	DISPL
6426 3E27	06120	LD	A, 27H
6428 C5	06130 DISPL	PUSH	BC
6429 D5	06140	PUSH	DE
642A E5	06150	PUSH	HL
642B 2A9B64	06160	LD	HL, (CLOCN)
642E FE0E	06170	CP	0EH
6430 2007	06180	JR	NZ, SPOTA
6432 3EBF	06190	LD	A, 0BFH
6434 329D64	06200	LD	(CCHAR), A
6437 1B33	06210	JR	CHANGC
6439 FE0F	06220 SPOTA	CP	0FH
643B 2007	06230	JR	NZ, SPOTB
643D 3E20	06240	LD	A, 20H
643F 329D64	06250	LD	(CCHAR), A
6442 1B28	06260	JR	CHANGC
6444 3620	06270 SPOTB	LD	(HL), 20H
6446 FE20	06280	CP	20H
6448 3B2A	06290	JR	C, CONTRL
644A 77	06300 SHOW	LD	(HL), A
644B 23	06310 ADV	INC	HL
644C CB74	06320 SCROCK	BIT	6, H

644E 2819	06330	JR	Z, SETCUR
6450 11003C	06340 SCROLL	LD	DE, 3C00H
6453 21403C	06350	LD	HL, 3C40H
6456 01C003	06360	LD	BC, 960
6459 EDB0	06370	LDIR	
645B 62	06380	LD	H, D
645C 6B	06390	LD	L, E
645D 13	06400	INC	DE
645E 014000	06410	LD	BC, 64
6461 212000	06420	LD	HL, 20H
6464 EDB0	06430	LDIR	
6466 21C03F	06440	LD	HL, 3FC0H
6469 3A9D64	06450 SETCUR	LD	A, (CCHAR)
646C 77	06460 CHANGC	LD	(HL), A
646D 229B64	06470	LD	(CLOCN), HL
6470 E1	06480	POP	HL
6471 D1	06490	POP	DE
6472 C1	06500	POP	BC
6473 C9	06510	RET	
6474 FE08	06520 CONTRL	CP	08H
6476 2008	06530	JR	NZ, SPOTC
6478 2B	06540	DEC	HL
6479 CB54	06550	BIT	2, H
647B 20EC	06560	JR	NZ, SETCUR
647D 23	06570	INC	HL
647E 18E9	06580	JR	SETCUR
6480 FE0A	06590 SPOTC	CP	0AH
6482 200B	06600	JR	NZ, SPOTD
6484 7D	06610 CRLF	LD	A, L
6485 E6C0	06620	AND	192
6487 6F	06630	LD	L, A
6488 1600	06640 LNFEED	LD	D, 0
648A 1E40	06650	LD	E, 40H
648C 19	06660	ADD	HL, DE
648D 18BD	06670	JR	SCROCK
648F FE0D	06680 SPOTD	CP	0DH
6491 28F1	06690	JR	Z, CRLF
6493 FE1A	06700	CP	1AH
6495 28F1	06710	JR	Z, LNFEED
6497 E1	06720	POP	HL
6498 D1	06730	POP	DE
6499 C1	06740	POP	BC
649A C9	06750	RET	
649B 003C	06760 CLOCN	DEFW	3C00H
649D 8F	06770 CCHAR	DEFB	0BFH
	06780 ;		
	06790 ;		
	06800 ;		
	06810 ;		
		KEYBOARD INPUT	
		ROUTINE	
649E C5	06820 SCAN	PUSH	BC
649F D5	06830	PUSH	DE
64A0 E5	06840	PUSH	HL

64A1 210138	06850	LD	HL, 3801H
64A4 0607	06860	LD	B, 7
64A6 7E	06870 SLOOP	LD	A, (HL)
64A7 B7	06880	OR	A
64A8 2006	06890	JR	NZ, FOUND
64AA CB15	06900	RL	L
64AC 10FB	06910	DJNZ	SLOOP
64AE 1835	06920	JR	FINE
64B0 01FFFF	06930 FOUND	LD	BC, OFFFFH
64B3 04	06940 CLOP	INC	B
64B4 CB1F	06950	RR	A
64B6 30FB	06960	JR	NC, CLOP
64B8 7D	06970	LD	A, L
64B9 0C	06980 RLOP	INC	C
64BA CB1F	06990	RR	A
64BC 30FB	07000	JR	NC, RLOP
64BE CB21	07010	SLA	C
64C0 CB21	07020	SLA	C
64C2 CB21	07030	SLA	C
64C4 78	07040	LD	A, B
64C5 B1	07050	OR	C
64C6 E5	07060	PUSH	HL
64C7 21F064	07070	LD	HL, TABLE
64CA 1600	07080	LD	D, 0
64CC 5F	07090	LD	E, A
64CD 19	07100 READY	ADD	HL, DE
64CE 4E	07110	LD	C, (HL)
64CF E1	07120	POP	HL
64D0 3A8038	07130	LD	A, (3880H)
64D3 CB47	07140	BIT	0, A
64D5 2802	07150	JR	Z, WLOOP
64D7 CBF9	07160	SET	7, C
64D9 7E	07170 WLOOP	LD	A, (HL)
64DA B7	07180	OR	A
64DB 20FC	07190	JR	NZ, WLOOP
64DD 0600	07200	LD	B, 0
64DF 00	07210 DLOOP	NOP	
64E0 00	07220	NOP	
64E1 00	07230	NOP	
64E2 10FB	07240	DJNZ	DLOOP
64E4 79	07250	LD	A, C
64E5 E1	07260 FINE	POP	HL
64E6 D1	07270	POP	DE
64E7 C1	07280	POP	BC
64E8 C9	07290	RET	
64E9 CD9E64	07300 INKEY	CALL	SCAN
64EC B7	07310	OR	A
64ED 28FA	07320	JR	Z, INKEY
64EF C9	07330	RET	
64F0 40	07340 TABLE	DEFM	' 0ABCDEFGHIJKLM'
64FE 4E	07350	DEFM	' NOPQRSTUVWXYZ'
650B 0000	07360	DEFW	0000

650D 0000	07370	DEFW	0000
650F 00	07380	DEFB	00
6510 30	07390	DEFM	'0123456789:;,.-./'
6520 0D	07400	DEFB	0DH
6521 1F	07410	DEFB	1FH
6522 01	07420	DEFB	01H
6523 5B	07430	DEFB	5BH
6524 0A	07440	DEFB	0AH
6525 08	07450	DEFB	08H
6526 09	07460	DEFB	09H
6527 20	07470	DEFB	20H
6528 0000	07480	DEFW	0000
652A 0000	07490	DEFW	0000
652C 0000	07500	DEFW	0000
652E 0000	07510	DEFW	0000
6530 0000	07520	DEFW	0000
6532 00	07530	NOP	
	07540 ;		
	07550 ;		
	07560 ;		
		DATA AND MESSAGE AREA	
6533 00	07570 NUM	DEFB	0
6534 53	07580 AMSG	DEFM	'SELECT 2-9'
653E 20	07590	DEFM	' GENERATORS*'
654A 45	07600 BMSG	DEFM	'ENTER YOUR RELATORS. '
655F 20	07610	DEFM	' LABEL YOUR '
656B 000A	07620 DOTA	DEFW	0A00H
656D 47	07630	DEFM	'GENERATORS WITH'
657C 20	07640	DEFM	' THE LETTERS A-'
658B 000A	07650 DOTB	DEFW	0A00H
658D 53	07660	DEFM	'SHIFT KEYS TO'
659A 20	07670	DEFM	' ENTER AN INVERSE/'
65AC 28	07680	DEFM	' (DISPLAYED WITH '
65BC 54	07690	DEFM	'TRAILING APOSTROPHY)//*'
65D3 48	07700 CMSG	DEFM	'HERE ARE YOUR RELATORS./'
65EB 54	07710	DEFM	'TO REENTER THE'
65F9 20	07720	DEFM	' ENTIRE SET TYPE R/'
660C 54	07730	DEFM	'TO CONTINUE TYPE C//*'
6621 41	07740 JMSG	DEFM	'ALEXANDER POLYNOMIAL: *'
6639 2F	07750 KMSG	DEFM	' /
6645 20	07760	DEFM	' *
6652 20	07770 LMSG	DEFM	' T'
6654 5B	07780	DEFB	5BH
6655 20	07790	DEFM	' *'
6657 45	07800 MMSG	DEFM	'END/STRIKE ANY'
6665 20	07810	DEFM	' KEY TO REPEAT*'
6674 49	07820 ZMSG	DEFM	'INVALID DATA OR FORMAT'
668A 0000	07830 PRDPTR	DEFW	0
0060	07840 RELS	DEFS	60H
0067	07850 BUFF	DEFS	67H
01D9	07860 POLY	DEFS	473
00C8	07870 PLYPRD	DEFS	200
0660	07880 MATRIX	DEFS	1632

6000 07890
00000 TOTAL ERRORS

END THESIS

VITA

Wallace G. Hynds III was born in Tampa, Florida, on March 4, 1957. He graduated from Gallatin Senior High School in June, 1975. The following September he entered The University of Tennessee, Knoxville, where he received a Bachelor of Arts degree in Mathematics.

In September, 1979, he entered the Graduate School at The University of Tennessee.