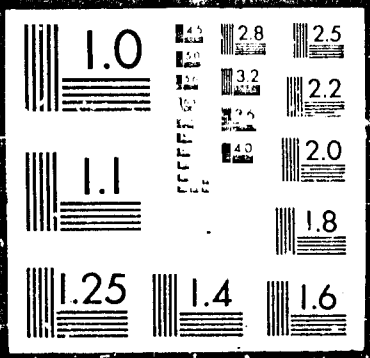
AD
740328



NATIONAL TECHNICAL
INFORMATION SERVICE

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Project MAC
Massachusetts Institute of Technology

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP
None

3. REPORT TITLE
Bounds on Polynomial Evaluation Algorithms

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Master of Science Thesis, Dept. of Electrical Engineering, January 1972

5. AUTHOR(S) *(Last name, first name, initial)*
Stockmeyer, Larry Joseph

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
April 1972	56	13

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-70-A-0362-0001	MAC TR-98
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

DDC
RECEIVED
APR 21 1972
A

10. AVAILABILITY/LIMITATION NOTICES
Distribution of this document is unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Advanced Research Projects Agency 3D-200 Pentagon Washington, D. C. 20301

13. ABSTRACT The purpose of this work is to investigate the number of arithmetic operations required by algorithms which evaluate polynomials. Previous results show that a polynomial of degree n requires at least n/2 multiplication/divisions and at least n addition/subtractions for its evaluation if the coefficients of the polynomial are suitably independent irrational numbers. However, the coefficients of any polynomial that would be evaluated in practice are represented only to a finite accuracy and are therefore rational numbers. The above results are extended to show that the same lower bounds hold for almost all rational polynomials if the polynomial is being evaluated efficiently. Another lower bound result is given that shows that almost all rational polynomials of degree n require at least $\sqrt{n}$ multiplication/divisions for their evaluation by any algorithm, efficient or not.

Several algorithms are presented which can in theory evaluate any rational polynomial using $0(\sqrt{n})$ multiplications and many additions. While of no practical use for rational polynomials in general, these algorithms do turn out to give methods for evaluating a polynomial at a matrix argument which are more efficient than previous methods.

14. KEY WORDS
polynomial evaluation, optimal algorithm, multiplications required, rational operations

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution

DD FORM 1 NOV 65 1473 (M.I.T.)

UNCLASSIFIED

Security Classification

ARPA Order Number 433
Program Code Number N66017
ARPA Contract Number N00014-70-A-0362-0001
Principal Investigator: Edward Fredkin
Phone Number: 617-864-6900 Ext. 5852
Name of Contractor: Massachusetts Institute of Technology
Effective Date of Contract: 1 March 1963
Contract Expiration Date: 30 June 1972
Amount of Contract: $26,601,248.00
Scientific Officer: The Scientific Officer under this Task Order is:
Marvin Denicoff
Director,
Information Systems Branch
Mathematical Sciences Division
Office of Naval Research
Dept. of the Navy
Arlington, Virginia 22217
Short Title of Work: ...Polynomial Evaluation Algorithms

This research was supported by the Advanced Research Projects Agency of the Dept. of Defense and was monitored by ONR under Contract No. N00014-70-A-0362-0001.

The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U. S. Government.

BOUNDS ON POLYNOMIAL EVALUATION ALGORITHMS

Larry Joseph Stockmeyer

April 1972

PROJECT MAC

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Cambridge Massachusetts

ABSTRACT

The purpose of this work is to investigate the number of arithmetic operations required by algorithms which evaluate polynomials. Previous results show that a polynomial of degree n requires at least n/2 multiplication/divisions and at least n addition/subtractions for its evaluation if the coefficients of the polynomial are suitably independent irrational numbers. However, the coefficients of any polynomial that would be evaluated in practice are represented only to a finite accuracy and are therefore rational numbers. The above results are extended to show that the same lower bounds hold for almost all rational polynomials if the polynomial is being evaluated efficiently. Another lower bound result is given that shows that almost all rational polynomials of degree n require at least $\sqrt{n}$ multiplication/divisions for their evaluation by any algorithm, efficient or not.

Several algorithms are presented which can in theory evaluate any rational polynomial using $O(\sqrt{n})$ multiplications and many additions. While of no practical use for rational polynomials in general, these algorithms do turn out to give methods for evaluating a polynomial at a matrix argument which are more efficient than previous methods.

ACKNOWLEDGMENT

I would like to thank Professors Albert R. Meyer, Michael S. Paterson, and Michael J. Fischer for many interesting discussions during the course of this work. It was they who initially formulated the question of looking at bounds on rational polynomial evaluation and who obtained the first results on which this work is based.

TABLE OF CONTENTS

I. Introduction.

One purpose of computational complexity is to find lower bounds on the degree of difficulty involved in performing some computation or class of computations under some measure of difficulty. Another aim is then to show that the lower bound is "tight" by exhibiting an algorithm which performs the computation with a degree of difficulty close to the lower bound.

This paper will consider the problem of evaluating a polynomial in one variable with real coefficients,

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 ,$$
$$a_0, a_1, \cdots, a_n \in P.$$

Polynomials arise very often in practice and many functions are evaluated by evaluating a finite portion of their Taylor polynomial.

The measure of difficulty will be the number of individual arithmetic operations +, -, *, ÷, used to evaluate the polynomial. We will be interested in how the required number of operations grows as a function of n, the degree of the polynomial. The greatest emphasis will be on counting the number of multiplications and divisions.

Algorithms will consist of a sequence of arithmetic operations. At each step, each of the two arguments for the operation may be some input variable, some fixed number,

or the result of some previous step. This is formalized in the following definition.

Definition. Let S be an infinite field and V be a finite set of input variables. A rational algorithm A over S is a sequence $A = a(1), a(2), \cdots, a(k)$ where

1). $a(1) \in S \cup V$ and

2). For $2 \leq r \leq k$ either $a(r) \in S \cup V$

or $a(r) = (op, i, j)$

where $1 \leq i, j < r$ and $op \in \{+, -, *, \div\}$

Let S(V) denote the extension field formed by attaching the indeterminates in V to the field S and closing under the rational operations.

Define the associated elements $p_1, p_2, \cdots, p_k \in S(V) \cup \{\infty\}$ as

$$p_r = \begin{cases} a(r) & \text{if } a(r) \in S \cup V \\ p_i \text{ op } p_j & \text{if } a(r) = (op, i, j) \text{ and } p_i \neq \infty, \\ & \text{and } p_j \neq \infty \text{ and either} \\ & op \neq \div \text{ or } p_j \not\equiv 0 \\ \infty & \text{otherwise} \end{cases}$$

Now A computes the function $f \in S(V)$ if

$$f = p_r \text{ for some } r, \quad 1 \leq r \leq k.$$

For example, to evaluate a polynomial p(x) with real coefficients, one could choose $S = R$ and $V = \{x\}$. (Throughout the remainder of this paper, C will denote the field of complex numbers, R will denote the field of real

numbers, Q the field of rational numbers, and Z the ring of integers).

Quite a bit of work has already been done concerning the complexity of polynomial evaluation algorithms. The new work presented here was initially motivated by considering polynomials with rational coefficients. The only known lower bound arguments show that at least n/2 multiplications are required to evaluate a degree n polynomial but assume that the coefficients $a_0, a_1, \cdots, a_n$ of the polynomial are algebraically independent real numbers, that is, there is no rational polynomial $P \in Q[y_0, y_1, \cdots, y_n]$, $P \not\equiv 0$, such that $P(a_0, a_1, \cdots, a_n) = 0$.

Such an assumption is necessary to obtain interesting lower bounds because some degree n polynomials, such as $p(x) = x^n + x^{n-1} + x^{n-2} + \cdots + x + 1$ can be evaluated in $O(\log n)$ operations.

If the coefficients of the polynomial are rational numbers, then they are necessarily algebraically dependent and the known lower bound methods do not directly apply. In fact, any rational polynomial of degree n can in theory be evaluated using only $O(\sqrt{n})$ multiplications as will be seen later. The coefficients of any polynomial that we would want to evaluate in practice would be represented only to a fixed number of decimal places and would therefore be rational numbers.

Before presenting the results for rational polynomials, the next section will give a brief survey of previous and current results concerning polynomial evaluation.

II. Survey of Previous Work.

Previous algorithms and lower bounds fall into two main classes, those which use preprocessing and those that do not. The latter class, no preprocessing, use the coefficients $a_0, a_1, \cdots, a_n$ of the polynomial being evaluated as the fixed numbers (scalars) which enter into the algorithm. The best known algorithm of this type is the scheme known as Horner's rule which evaluates

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \quad \text{as}$$

Algorithm A.

$$p(x) = (((\cdots(a_n x + a_{n-1})x + a_{n-2})x + \cdots + a_1)x + a_0$$

in n multiplications and n additions.

Preprocessing algorithms use as scalars certain real numbers which are precomputed from the a_i. There are preprocessing algorithms which use about n/2 multiplications and n additions. These algorithms are most useful when the polynomial is to be evaluated many times at different points. The preprocessing need only be done once and n/2 multiplications are then saved over Horner's rule each time the polynomial is evaluated.

2.1 Preprocessing.

A preprocessing algorithm is presented first.

Theorem 2.1 (Motzkin [1], Pan [2]). Any degree n real polynomial can be evaluated using $\lfloor n/2 \rfloor + 2$ multiplications and n additions.

Proof:

Algorithm B. $p(x) = a_n x^n + \cdots + a_1 x + a_0$ is evaluated by the scheme

$$y = x + c \qquad w = y^2$$

$$z = (a_n y + s_0)y + t_0 \quad (n \text{ even})$$

$$z = a_n y + t_0 \quad (n \text{ odd})$$

$$p(x) = (((\cdots(z(w - s_1) + t_1)(w - s_2) + t_2) \cdots)(w - s_m)$$

where $m = \lceil n/2 \rceil - 1$

and c, s_i, t_i $(i = 0, \cdots, m)$ are certain real numbers which are found in the following manner.

First note that $\deg(p) = n$ implies that p can be written as $p(x) = (x^2 - s_m) \cdot q(x) + t_m$

where $\deg(q) = n - 2$ and $s_m, t_m \in C$.

If $p(x) = a_n x^n + \cdots + a_1 x + a_0$, then s_m is a root of the auxiliary polynomial formed from p by taking each "odd" coefficient $p_{AUX}(x) = a_{2m+1} x^m + \cdots + a_3 x + a_1$.

t_m is found by a polynomial division. Now apply this to q(x), finding a s_{m-1} and t_{m-1}, and continue. It turns out that $s_1, \cdots, s_m$ are the m roots of $p_{aux}(x)$ and the t_i are obtained from the s_i and a_i by polynomial divisions. Note

that the s_i and t_i may be complex even though the a_i are real. this is undesirable because complex arithmetic requires at least twice as many real operations as real arithmetic and any advantage over a non-preprocessing algorithm is lost. However J. Eve [3] has noted that

if $\hat{p}(y) = p(y - c) = p(x)$,

then all the roots of $\hat{p}_{aux}(y)$ are real for an appropriate real c. ___/

For a further discussion of Algorithm B, see Knuth [4].

Turning now to the problem of finding a lower bound, it is possible to show that at least $\lceil n/2 \rceil$ multiplication/divisions are required to evaluate a degree n polynomial if the coefficients are algebraically independent real numbers. A multiplication/division operation is either a multiplication or a division. It is instructive to view this proof in some detail as the methods will be extended and used later. First, the concept of algebraic independence is defined.

Definition. $a_1, \cdots, a_n \in R$ are said to be algebraically independent if there is no $P \in Q[y_1, \cdots, y_n]$, $P \not\equiv 0$, such that $P(a_1, \cdots, a_n) = 0$.

The first lemma shows the maximum amount of computation that can be done with a certain number of multiplication/divisions.

Lemma 2.1. A polynomial p(x) can be evaluated in no more than k multiplication/divisions by a rational algorithm over R if and only if it can be evaluated by the scheme S_k given by

$$u_1 = (m_{10}x + s_1)\ op_1\ (\hat{m}_{10}x + \hat{s}_1)$$

$$u_2 = (m_{21}u_1 + m_{20}x + s_2)\ op_2\ (\hat{m}_{21}u_1 + \hat{m}_{20}x + \hat{s}_2)$$

in general

$$u_r = \left(\sum_{i=1}^{r-1} m_{ri}u_i + m_{r0}x + s_r\right)\ op_r\ \left(\sum_{i=1}^{r-1} \hat{m}_{ri}u_i + \hat{m}_{r0}x + \hat{s}_r\right) \qquad r = 2, \cdots, k$$

and finally $p(x) = \sum_{i=1}^{k} m_{k+1,i}u_i + m_{k+1,0}x + s_{k+1}$

where $m_{ij}, \hat{m}_{ij} \in Z$ for all i, j

$s_i, \hat{s}_i \in R$ for all i

$op_i \in \{*, \div\}$ for all i

Proof : The proof is complete once it is noticed that after r-1 multiplication/divisions have been performed, the only computation that can be performed without doing another multiplication/division is addition/subtractions on

$$\{x, u_1, u_2, \cdots, u_{r-1}\} \cup R$$

where u_i is the result of the i-th multiplication/division. Any such computation can be written as

$$\sum_{i=1}^{r-1} m_{ri}u_i + m_{r0}x + s_r$$

where $m_{ri} \in Z \quad i = 0, \cdots, r-1$ and $s_r \in R$.

The integer "multiplication" $m_{ri}u_i$ is just shorthand for

m_{ri} repeated additions of u_i.

Two terms of this type can be formed and combined with a multiplication or division to get the next multiplication/division step u_r. ___/

The next lemma is also necessary.

Lemma 2.2 Suppose $p_i \in Q(x_1, \cdots, x_m)$ $i = 1, \cdots, n$. That is, there are n rational functions, each in the m variables $x_1, \cdots, x_m$. If $n > m$, then there exists a multivariate rational polynomial $P \in Q[y_1, \cdots, y_n]$, $P \not\equiv 0$, such that $P(p_1(x_1, \cdots, x_m), \cdots, p_n(x_1, \cdots, x_m)) = 0$ for all $x_1, \cdots, x_m$ such that $p_i \neq \infty$, $i = 1, \cdots, n$. Informally, if there are more functions than variables, then they satisfy a non-trivial polynomial relation.

A formal proof of this lemma will not be given. Very informally, if $p_1, \cdots, p_n$ were algebraically independent, then $Q(p_1, \cdots, p_n)$ would have degree of transcendence n over Q. But since the p_i are rational functions in the x_i, $Q(p_1, \cdots, p_n) \subseteq Q(x_1, \cdots, x_m)$ which has degree of transcendence m over Q. $n > m$ gives the contradiction. For further details see for example [5]. ___/

The following lower bound result can now be proven.

Theorem 2.2 (Motzkin[1], Winograd [6]).
Any $p(x) = a_n x^n + \cdots + a_1 x + a_0$ with $a_0, a_1, \cdots, a_n$ algebraically independent real numbers requires at least $\lceil n/2 \rceil$ multiplication/divisions for its evaluation by any rational algorithm over R.

Proof : Assume p(x) can be evaluated in k multiplication/divisions and therefore by the scheme S_k for some choice of $m_{ij}, \hat{m}_{ij} \in Z, \quad s_i, \hat{s}_i \in R, \quad op_i \in \{*, \div\}$.
Formally carry out the operations in S_k and view the result as a polynomial in x,

$$p(x) = p_n(\vec{s})x^n + \cdots + p_1(\vec{s})x + p_0(\vec{s})$$
$$= a_n x^n + \cdots + a_1 x + a_0 \ ,$$

where each p_i is a rational function in

$$\vec{s} = (s_1, \hat{s}_1, \cdots, s_k, \hat{s}_k, s_{k+1}).$$

Note that there are $2k + 1$ variables in $\vec{s}$.
Now if $2k + 1 < n + 1$, then by Lemma 2.2 there is a rational polynomial $P \not\equiv 0$ such that

$$P(p_n(\vec{s}), \cdots, p_0(\vec{s})) = 0$$

or $P(a_n, \cdots, a_1, a_0) = 0$ which is a contradiction.
Therefore $2k + 1 \geq n + 1$ or $k \geq n/2$. ___/

It is useful to think of the $s_i, \hat{s}_i$ as representing degrees of freedom. The set of all degree n polynomials has n + 1 degrees of freedom because each coefficient can be varied independently. Lemma 2.1 states that each multiplication/division can introduce at most two degrees of freedom into the algorithm. The concept of degrees of freedom is formalized in

Knuth [4], but the method of proof of Theorem 2.2 using degrees of freedom is essentially the one presented here.

Using similar techniques, it can be shown that

Theorem 2.3 (Belaga [7]). Any $p(x) = a_n x^n + \cdots + a_1 x + a_0$ with $a_0, a_1, \cdots, a_n$ algebraically independent real numbers requires at least n addition/subtractions for its evaluation by any rational algorithm over R.

A proof is also presented in Knuth [4] . It is done by showing that each addition/subtraction can introduce at most one degree of freedom except for a single addition/subtraction which can introduce two. ___/

Therefore Algorithm B is an almost optimal (± two multiplications) method for polynomial evaluation with preprocessing.

2.2 Rational Preprocessing

Part of the preprocessing required by Algorithm B involves finding all the roots of a polynomial of degree n/2. This may be computationally difficult in itself and may lead to inaccuracies in the actual evaluated values of the polynomial. The latter is true because even if the coefficients of the polynomial being evaluated are rational numbers and have a finite decimal representation, the scalars used by the algorithm may be irrational numbers which cannot be represented as their exact value but must be rounded to a

finite number of decimal places. The following algorithm, discovered by Michael Paterson, has the advantage that the preprocessing involves only the rational arithmetic operations. This advantage is obtained at the cost of doing a few more multiplications during the actual evaluation of the polynomial.

Theorem 2.4 (Paterson). Any degree n real polynomial can be evaluated using $n/2 + O(\log n)$ multiplications. Moreover, the scalars used by the algorithm are rational functions of the coefficients of the polynomial.

Proof :

Algorithm C. For the moment assume p(x) is monic (that is, $a_n = 1$) and $\deg(p) = n = 2^j - 1$ for some positive integer j. First compute $x^2, x^4, x^8, \cdots, x^{2^{j-1}}$ in $\lfloor \log n \rfloor$ multiplications. (all logarithms are taken to base 2)

Now note that if p(x) is monic and $\deg(p) = 2m - 1$, then p(x) can be written as

$$p(x) = (x^m + c)q(x) + r(x)$$

where q and r are monic, $\deg(q) = \deg(r) = m - 1$, and the coefficients of q and r are given by rational functions of the coefficients of p.

In fact $x^{2m-1} + a_{2m-2}x^{2m-2} + \cdots + a_1x + a_0$

$$= (x^m + c)(x^{m-1} + a_{2m-2}x^{m-2} + \cdots + a_{m+1}x + a_m) + (x^{m-1} + b_{m-2}x^{m-2} + \cdots + b_1x + b_0)$$

where $c = a_{m-1} - 1$

and $b_i = a_i - ca_{m+i}$ $\quad i = 0, \cdots, m-2.$

If $\deg(p) = 2m - 1 = 2^j - 1,$

then $\deg(q) = \deg(r) = 2^{j-1} - 1$

and $x^m = x^{2^{j-1}}$ is one of the powers that were computed at the beginning. $q(x)$ and $r(x)$ are of the proper form (monic of degree $2^i - 1$ for some i) and the procedure may be applied recursively to them.

Let $M(n)$ = the number of multiplications required to evaluate a monic degree n polynomial by this procedure, assuming that $x^2, x^4, x^8, \cdots, x^{2^{\lfloor \log n \rfloor}}$ have been computed. Then by the above argument,

$$M(2^i - 1) = 2M(2^{i-1} - 1) + 1 \qquad i = 2, 3, \cdots j.$$

Also $M(1) = 0$ because $x + a_0$ can be evaluated using no multiplications. This recurrence relation solves as

$$M(2^i - 1) = 2^{i-1} - 1 \qquad i = 1, 2, 3, \cdots, j.$$

So $M(n) = M(2^j - 1) = 2^{j-1} - 1 = (n + 1)/2 - 1.$

Allowing one more multiplication for the monic division,

$$M(n) = (n + 1)/2.$$

Total multiplications $= (n + 1)/2 + \lfloor \log n \rfloor$ if $n = 2^j - 1.$

For general n, the polynomial can be broken into pieces, each of degree $2^i - 1$ for some i. The pieces can be evaluated separately and put back together using the powers $x^2, x^4, x^8, \cdots$. The putting-back-together can require at most another $\log n$ multiplications for a total of about

$$n/2 + 2\log n. \qquad \_\_\_/$$

The number of additions used by Algorithm C can also be counted.

Let $A(n)$ = the number of additions required by Algorithm C to evaluate a degree n polynomial.

A satisfies the recurrence relation

$$A(2^i - 1) = 2A(2^{i-1} - 1) + 2 \qquad i = 2,3,4, \cdots$$

$$A(1) = 1$$

which solves as

$$A(2^i - 1) = 2^i + 2^{i-1} - 2 = (3/2)\cdot 2^i - 2 \qquad i = 1,2,3,\cdots$$

$$\text{or} \quad A(n) \approx (3/2)n.$$

Therefore, Algorithm C uses $2n + O(\log n)$ arithmetic operations, and is no more efficient than Horner's rule in this respect. However, if multiplication speed is slower than addition speed, Algorithm C is more efficient than non-preprocessing schemes such as Horner's rule if the polynomial is to be evaluated many times.

2.3 No Preprocessing

Since an algorithm without preprocessing must start from the a_i themselves as scalars, it must in effect evaluate the general polynomial

$$P(x, a_0, a_1, \cdots, a_n) = a_n x^n + \cdots + a_1 x + a_0$$

where the a_i as well as x are inputs. The following result shows that Horner's rule is an optimal algorithm for P.

Theorem 2.5 (Pan [2], Winograd[6]). Any algorithm which computes $P(x, a_0, a_1, \cdots, a_n) = a_n x^n + \cdots + a_1 x + a_0$ requires at least n multiplication/divisions.

No proof will be given.

Borodin [8] has shown that Horner's rule is the only algorithm which evaluates $P(x, a_0, a_1, \cdots, a_n)$ in n multiplications and n additions. Therefore, Horner's rule is a unique optimal method for polynomial evaluation with no preprocessing.

2.4 Parallel Algorithms

In previous sections, algorithms were assumed to be sequential. Work has been done concerning algorithms which can do many operations at each step and some of the results will be considered next without proofs.

Parallel algorithms can be formalized as follows. At step r, the algorithm may compute m terms of the form

$$u_{ri} = s_i \; op_i \; t_i \qquad i = 1, \cdots, m$$

where $op_i \in \{+, -, *, \div\}$ and each of s_i and t_i are some input variable, some fixed number, or one of the results of some previous step j, $1 \leq j < r$. If only k processors are available, then $m \leq k$ at each step. All arithmetic operations are given equal weight. The most interesting results concern the problem of evaluating the general polynomial $P(x, a_0, a_1, \cdots, a_n) = a_n x^n + \cdots + a_1 x + a_0$

with no preprocessing.

Estrin [9] has given a parallel algorithm which has the merit of simplicity of description. The algorithm computes $p(x)$ of degree n as $p(x) = q(x)\cdot x^{n/2+1} + r(x)$, where $\deg(q) = \deg(r) = n/2$, and then q and r similarly by a binary splitting, and so on. Thus it starts by computing

$$x^2,\ a_1x + a_0,\ a_3x + a_2,\ \cdots,\ a_nx + a_{n-1}$$

in the first two steps, then

$$x^4,\ (a_3x + a_2)x^2 + (a_1x + a_0),\ \cdots$$ in the next two, etc. If an unlimited number of processors are available, this scheme requires about $2\log n$ steps.

Dorn [10] has modified this to obtain an algorithm which uses only $k < n$ processors and runs in

$$2n/k + 2\log k \quad \text{steps.}$$

Recent work by Munro and Paterson has improved these algorithms to reduce the coefficient of the log term to 1. They also give lower bound results.

Theorem 2.6 (Munro, Paterson [11]). Any parallel algorithm which computes the general polynomial of degree n requires at least $\lceil \log n \rceil + 1$ steps. If only k processors are available, then $\lceil 2n/k \rceil + \lceil \log k \rceil - 1$ steps are required.

They give algorithms which run in time closer to these lower bounds than previous results.

Theorem 2.7 (Munro, Paterson[11]). The general polynomial of degree n can be evaluated in $\log n + O(\sqrt{\log n})$ steps by a parallel algorithm using n processors. If only k processors are available, $O(\log n) \leq k < n$, then the evaluation can be done in $2n/k + \log k + O(\sqrt{\log k})$ steps.

III. Main Results

This section will be concerned with finding lower bounds on algorithms which evaluate rational polynomials and exhibiting algorithms which can in theory be used to evaluate any rational polynomial using a number of multiplications close to the lower bound. Since any set of rational numbers are necessarily algebraically dependent, the lower bound results of section II are not directly applicable. In fact, one might suspect that rational polynomials of degree n could be evaluated in fewer than $n/2$ multiplications because an integer multiplication can be done free of multiplication by doing many additions, $z \cdot u = u + u + \cdots + u$ (z times). Any rational polynomial $r(x) \in Q[x]$ is of the form

$$r(x) = r_0 \cdot z(x) \quad \text{where} \quad r_0 \in Q, \quad z(x) \in Z[x].$$

Because of this fact, it is useful to single out those multiplications of the form $c \cdot u$ where c is a fixed number (that is, c has no dependence on x).

Definition. Referring to the definition of a rational algorithm over a scalar field S, the step $a(r)$ defines a scalar multiplication if

$a(r) = (*, i, j)$ and either $a(i) \in S$ or $a(j) \in S$.

$a(r)$ defines a scalar division if

$$a(r) = (\div, i, j) \quad \text{and} \quad a(j) \in S.$$

Any multiplication or division which is not scalar will

be called non-scalar. Non-scalar multiplications are of the form $q(x)\cdot r(x)$ and cannot be eliminated by successive additions.

3.1 Lower Bounds

The first lemma shows the maximum amount of computation that can be done with a certain number of non-scalar multiplication/divisions.

Lemma 3.1. A polynomial $p(x)$ can be evaluated in no more than k non-scalar multiplication/divisions by a rational algorithm over S if and only if it can be evaluated by the scheme A_k given by

$$v_1 = (m_{10}x + m_{1,-1}) \; op_1 \; (\hat{m}_{10}x + \hat{m}_{1,-1})$$

$$u_2 = (m_{21}u_1 + m_{20}x + m_{2,-1}) \; op_2 \; (\hat{m}_{21}u_1 + \hat{m}_{20}x + \hat{m}_{2,-1})$$

in general

$$u_r = \left(\sum_{i=1}^{r-1} m_{ri}u_i + m_{r0}x + m_{r,-1}\right) op_r \left(\sum_{i=1}^{r-1} \hat{m}_{ri}u_i + \hat{m}_{r0}x + \hat{m}_{r,-1}\right)$$

$$r = 2,3,\cdots,k$$

and finally

$$p(x) = \sum_{i=1}^{k} m_{k+1,i}u_i + m_{k+1,0}x + m_{k+1,-1}$$

where $m_{ij}, \hat{m}_{ij} \in S$ for all i, j.

$op_i \in \{*, \div\}$ for all i

Proof : After r-1 non-scalar multiplication/divisions have been performed, the only computation that can be performed without doing another non-scalar multiplication/division is

addition/subtractions and scalar multiplication/divisions on

$$\{x, u_1, u_2, \cdots, u_{r-1}\} \cup S$$

Any such computation can be written as

$$\sum_{i=1}^{r-1} m_{ri} u_i + m_{r0} x + m_{r,-1} \quad \text{where} \quad m_{ri} \in S, \; i = -1, \cdots, r-1$$

Two terms of this type can be formed and combined with a multiplication or division to get the next non-scalar multiplication/division step u_r. ___/

The m_{ij} will be called the <u>parameters</u> of the algorithmic scheme A_k.

For what follows, it will be useful to count the number of parameters in A_k. The expression for u_r introduces $2r + 2$ parameters and the expression for $p(x)$ introduces $k + 2$ for a total of $\sum_{r=1}^{k} (2r + 2) + k + 2 = k^2 + 4k + 2$.

The first theorem digresses for a moment to consider polynomials with algebraically independent coefficients in the context of counting only non-scalar multiplication/divisions.

<u>Theorem 3.1</u> (M. Paterson). Any $p(x) = a_n x^n + \cdots + a_1 x + a_0$ with $a_0, a_1, \cdots, a_n$ algebraically independent real numbers requires at least $\lceil \sqrt{n+3} \rceil - 2$ non-scalar multiplication/divisions for its evaluation by any rational algorithm over R.

<u>Proof</u> : Assume $p(x)$ can be evaluated in k non-scalar multiplication/divisions and therefore by the scheme A_k for some choice of $m_{ij}, \hat{m}_{ij} \in R$, $\circ p_i \in \{*, \div\}$.

Formally carry out the operations in A_k and view the result as a polynomial in x,

$$p(x) = p_n(\vec{m})x^n + \cdots + p_1(\vec{m})x + p_0(\vec{m})$$
$$= a_n x^n + \cdots + a_1 x + a_0 ,$$

where each p_i is a rational function in $\vec{m}$, the vector of parameters of length $k^2 + 4k + 2$.

If $k^2 + 4k + 2 < n + 1$, then by Lemma 2.2, there is a rational polynomial $P \not\equiv 0$ such that

$$P(p_0(\vec{m}), \cdots, p_n(\vec{m})) = 0$$

or $P(a_0, \cdots, a_n) = 0$. Contradiction.

Therefore $k^2 + 4k + 2 \geq n + 1$ or $k \geq \sqrt{n + 3} - 2$. ___/

Lower bound results for rational polynomials will now be presented. It is difficult to obtain interesting lower bounds for all rational polynomials because some degree n rational polynomials can be evaluated fast in $O(\log n)$ operations. However, the lower bounds will be shown to apply to "almost all" rational polynomials. A set of rational vectors will satisfy the intuitive notion of being almost all rational vectors if all vectors not in the set satisfy a non-trivial rational polynomial relation.

<u>Definition</u>. A set $S \subseteq \{q(x) \in Q[x] \mid \deg(q) \leq n\}$ will be said to contain <u>almost all rational polynomials of degree n</u> if there is a $P \in Q[y_n, \cdots, y_0]$, $P \not\equiv 0$, such that $q_n x^n + \cdots + q_1 x + q_0 \in Q[x] - S$ implies $P(q_n, \cdots, q_0) = 0$.

In particular, if S contains almost all rational polynomials of degree n, then $S \neq \emptyset$.

If $S = \emptyset$ then $q_n x^n + \cdots + q_1 x + q_0 \in Q[x] - S$ for all $(q_n, \cdots, q_0) \in Q^{n+1}$ which implies that $P(Q^{n+1}) \equiv 0$. But since the rationals are dense in the reals and since P is continuous, this implies $P(R^{n+1}) \equiv 0$ or $P \equiv 0$ contrary to assumption.

A similar argument shows that if S contains almost all rational polynomials of degree n, then

$$\left\{ (q_n, \cdots, q_0) \in Q^{n+1} \;\middle|\; q_n x^n + \cdots + q_1 x + q_0 \in S \right\}$$

is a dense subset of R^{n+1}.

The first result assumes that the algorithm contains no divisions.

Theorem 3.2 For any $n > 0$, there are rational polynomials of degree n which require $\lceil \sqrt{n+3} \rceil - 2$ multiplications for their evaluation by any rational algorithm over R without divisions. In fact, almost all rational polynomials of degree n require $\lceil \sqrt{n+3} \rceil - 2$ multiplications.

Proof : Suppose $q(x) = q_n x^n + \cdots + q_1 x + q_0 \in Q[x]$ can be evaluated in k non-scalar multiplications and no divisions and therefore by the scheme A_k for $op_i = *$, $i = 1, \cdots, k$ and for some choice of $m_{ij}, \hat{m}_{ij} \in R$.

Formally carry out the operations in A_k and obtain $q(x) = p_n(\vec{m})x^n + \cdots + p_1(\vec{m})x + p_0(\vec{m}) = q_n x^n + \cdots + q_1 x + q_0$

where the p_i are rational polynomials in the parameters $\vec{m}$. Assume $k^2 + 4k + 2 < n + 1$ and find a rational polynomial $P \not\equiv 0$ such that $P(p_n(\vec{m}), \cdots, p_0(\vec{m})) = 0$

$$\text{or } P(q_n, \cdots, q_0) = 0.$$

Note that the $p_i(\vec{m})$ and P depend only on the form A_k, not on the particular polynomial being evaluated.

If all rational polynomials can be evaluated in no more than k non-scalar multiplications, then

$$P(q_n, \cdots, q_0) = 0 \quad \text{for all} \quad (q_n, \cdots, q_0) \in Q^{n+1}$$

But since the rationals are dense in the reals and since P is a continuous function, this implies

$$P(r_n, \cdots, r_0) = 0 \quad \text{for all} \quad (r_n, \cdots, r_0) \in R^{n+1}$$

$$\text{or} \quad P \equiv 0 \quad \text{contrary to assumption.}$$

Therefore, there must be some rational polynomials which require at least k non-scalar multiplications and therefore at least k multiplications where

$$k^2 + 4k + 2 \geq n + 1 \quad \text{or} \quad k \geq \sqrt{n + 3} - 2.$$

The second conclusion of the theorem (which actually implies the first) follows from the fact that

$P(q_n, \cdots, q_0) \neq 0$ implies that $q_n x^n + \cdots + q_1 x + q_0$ requires at least $\sqrt{n + 3} - 2$ multiplications where $P \not\equiv 0$ is as above. ___/

This lower bound can be improved slightly by noticing that the parameters in A_k must contain some redundancy.

Lemma 3.2 If $p(x)$ can be computed by the scheme A_k with no divisions, then $p(x)$ can be computed by A_k with

i). $m_{r,-1} = \hat{m}_{r,-1} = 0 \qquad r = 1,2,\cdots,k$

ii). For all r, $1 \le r \le k$, there is an $i \ge 0$ and $j \ge 0$ such that $m_{ri} = \hat{m}_{rj} = 1$

iii). $m_{20} = 0$.

Proof : i). Sequentially for $r = 1,2,\cdots,k$ write

$$u_r = (\sum_{i=1}^{r-1} m_{ri}u_i + m_{r0}x + m_{r,-1})*(\sum_{i=1}^{r-1} \hat{m}_{ri}u_i + \hat{m}_{r0}x + \hat{m}_{r,-1})$$

as $$u_r = (\sum_{i=1}^{r-1} m_{ri}u_i + m_{r0}x)*(\sum_{i=1}^{r-1} \hat{m}_{ri}u_i + \hat{m}_{r0}x) + \sum_{i=1}^{r-1} c_{ri}u_i + c_{r0}x + c_{r,-1}$$

where the c_i are scalars obtained from the m_{ri}, $\hat{m}_{ri}$.

Compute instead $u_r' = (\sum_{i=1}^{r-1} m_{ri}u_i + m_{r0}x)*(\sum_{i=1}^{r-1} \hat{m}_{ri}u_i + \hat{m}_{r0}x)$

and adjust the coefficients of $1, x, u_1, u_2, \cdots, u_{r-1}$ in all following steps to compensate for the lost terms

$$\sum_{i=1}^{r-1} c_{ri}u_i + c_{r0}x + c_{r,-1}$$

ii). For $r = 1,2,\cdots,k$, in step u_r, if

$m_{ri} \neq 0$ and $\hat{m}_{rj} \neq 0$,

compute instead $u_r' = (1/m_{ri}\hat{m}_{rj})\cdot u_r$

and adjust coefficients of u_r in all following steps.

iii). After the above two reductions have been made,

$$u_1 = x^2$$

$$u_2 = (x^2 + m_{20}x)*(x^2 + \hat{m}_{20}x) \quad \text{or} \quad u_2 = (x^2 + m_{20}x)*(x).$$

In the first case, compute instead

$$u_2' = (x^2)*(x^2 + (m_{20} + \hat{m}_{20})x)$$

and adjust the coefficient of u_1 in all following steps to compensate for the lost term $m_{20}\hat{m}_{20}x^2$.

In the second case, compute instead $u_2' = x^2 \cdot x$. ___/

Corollary 3.2 The lower bound of Theorem 3.2 can be raised to $\lceil \sqrt{n} \rceil$ multiplications required.

Proof : After the reductions of Lemma 3.2 have been made, there are $k^2 + 1$ non-constant parameters in A_k and $k^2 + 1 \geq n + 1$ gives $k \geq \sqrt{n}$. ___/

A similar result can be proven using combinatorial techniques under the assumption that an integer polynomial is being evaluated by an algorithm over Z. (Since there are no divisions, S need only be a ring). The proof is due to Michael Paterson.

Theorem 3.3 (Paterson). For any $n > 0$, there are integer polynomials of degree n which require at least $\lceil \sqrt{n} \rceil - 1$ multiplications for their evaluation by any algorithm over Z without divisions.

Proof : Consider the finite ring $F = \{0, 1\}$ and the ring homomorphism $H : Z \rightarrow F$ by

$$H(z) = \begin{cases} 1 & \text{if } z \text{ is odd} \\ 0 & \text{if } z \text{ is even} \end{cases}$$

First it is shown that if $z_n x^n + \cdots + z_1 x + z_0 \in Z[x]$ can be computed in k non-scalar multiplications by an algorithm over Z, then $w_n x^n + \cdots + w_1 x + w_0 \in F[x]$

where $w_i = H(z_i) \qquad i = 0,1,\cdots,n$,

can be computed in k non-scalar multiplications by an algorithm over F.

Formally carry out the operations in A_k (with no divisions) and obtain $p(x) = p_n(\vec{m})x^n + \cdots + p_1(\vec{m})x + p_0(\vec{m})$ where each p_i is a rational polynomial in $\vec{m}$.

Now $z_n x^n + \cdots + z_1 x + z_0$ computed by A_k over Z implies there is an $\vec{m} \in Z^{k'}$, where k' = number of parameters in A_k , such that $z_i = p_i(\vec{m}) \qquad i = 0,1,\cdots,n$

which implies $w_i = H(z_i) = p_i^*(H(\vec{m})) \qquad i = 0,1,\cdots,n$

where p_i^* means "do the arithmetic mod 2" and $H(\vec{m})$ is the vector in $F^{k'}$ obtained by applying H to each element of $\vec{m}$. This implies that $w_n x^n + \cdots + w_1 x + w_0$ is computed by A_k over F.

Note that reduction 1) of Lemma 3.2 can be done if S = Z, so that $k' = k^2 + 2k + 2$.

Assume $k^2 + 2k + 2 < n + 1$.

The proof will be complete if we can present polynomials of degree n in F[x] which cannot be computed by A_k over F. There are such polynomials because there are 2^{n+1} polynomials in F[x], but only 2^{k^2+2k+2} different polynomials can be computed by A_k over F.

Therefore $k^2 + 2k + 2 \geq n + 1$ or $k \geq \sqrt{n} - 1$. ___/

Now divisions are allowed in the algorithm and the same lower bound is shown to hold.

Theorem 3.4 For any $n > 0$, almost all rational polynomials of degree n require $\lceil \sqrt{n} \rceil - 1$ multiplication/divisions for their evaluation by any rational algorithm over R.

Proof : Note that reduction ii) of Lemma 3.2 can be done if the algorithm contains divisions, so that A_k contains $k^2 + 2k + 2$ parameters. Assume $k^2 + 2k + 2 < n + 1$.

Consider each of the 2^k algorithmic forms obtained from A_k by independently choosing a * or ÷ at each non-scalar step. Using techniques of Theorem 3.2, for each of these forms find a $P_i \in Q[y_n, \cdots, y_0]$, $P_i \not\equiv 0$, such that if $q_n x^n + \cdots + q_1 x + q_0$ is computed by the i-th form, then

$$P(q_n, \cdots, q_0) = 0.$$

Define $P(y_n, \cdots, y_0) = \prod_{i=1}^{2^k} P_i(y_n, \cdots, y_0)$.

$P \in Q[y_n, \cdots, y_0]$ and $P \not\equiv 0$.

Also, if $q_n x^n + \cdots + q_1 x + q_0$ can be computed in k non-scalar multiplication/divisions, then it can be computed by the i_0-th form, for some i_0, so that

$$P_{i_0}(q_n, \cdots, q_0) = 0 \quad \text{and} \quad P(q_n, \cdots, q_0) = 0.$$

Therefore $P(q_n, \cdots, q_0) \neq 0$ implies that $q_n x^n + \cdots + q_1 x + q_0$ requires at least k (non-scalar) multiplication/divisions where $k^2 + 2k + 2 \geq n + 1$.

The conclusion follows. ___/

Lower bound results will be returned to later in the context of bounded additions.

Several algorithms will be presented next which are optimal to within a constant multiple of the number of non-scalar multiplications required.

3.2 Algorithms

This section will present several algorithms which can evaluate any degree n polynomial using $O(\sqrt{n})$ non-scalar multiplications and no divisions. The algorithms are applicable to any real polynomial with in general algebraically independent coefficients.

Two of these algorithms can in theory be used to evaluate any rational polynomial in $O(\sqrt{n})$ total multiplications because if $q(x) \in Q[x]$ can be evaluated by an algorithm over Q, then $z_0 \cdot q(x) \in Z[x]$ can be evaluated by an algorithm over Z using the same number of non-scalar multiplications, for some $z_0 \in Z$. This is true because, for $r = 1,2,\cdots,k$ if $u_r = (\sum_{i=1}^{r-1} m_{ri} u_i + m_{r0} x) * (\sum_{i=1}^{r-1} \hat{m}_{ri} u_i + \hat{m}_{r0} x)$ with $m_{ri}, \hat{m}_{ri} \in Q$, compute instead $u'_r = z_r \cdot u_r$ where z_r is the least common multiple of the denominators of the $m_{ri}, \hat{m}_{ri}$, and adjust the coefficients of u_r in all following steps.

If the algorithm uses no preprocessing or rational preprocessing, then rational coefficients produce rational

scalars in the algorithm and the above procedure may be used to yield an algorithm with integer scalars. Therefore, scalar multiplications can be done by many additions and only non-scalar multiplications need be counted. The point is not that one would ever want to do this in practice, but that it would be impossible to obtain a stronger than $O(\sqrt{n})$ multiplications required lower bound for rational polynomials if rational polynomials can indeed be evaluated in $O(\sqrt{n})$ multiplications using this trick. A following section will bound the additions and show that a stronger lower bound holds.

A more practical use for these algorithms concerns the problem of evaluating a real or rational polynomial at a matrix argument,

$$p(A) = a_n A^n + \cdots + a_1 A + a_0 I$$

with $a_0, \cdots, a_n \in R$ and A a real matrix. For example, one might want to evaluate such a polynomial if evaluating e^{At} by a Taylor polynomial, where A is a matrix, in order to find the solution of a system of linear differential equations. In this case, a non-scalar (matrix)·(matrix) multiplication is slower than a scalar (number)·(matrix) multiplication (using state-of-the-art matrix multiplication procedures) and it would be useful to minimize the number of non-scalar multiplications.

3.2.1 Description of the Algorithms

The first algorithm is an "extended Horner's rule".

Theorem 3.5 (M. J. Fischer, A. R. Meyer, M. S. Paterson). Any degree n polynomial can be evaluated using $2\lceil\sqrt{n}\rceil - 2$ non-scalar multiplications and no divisions. No preprocessing of coefficients is required.

Proof :

Algorithm D. First compute $x^2, x^3, x^4, \ldots, x^k$ for some k, using $k - 1$ multiplications.

Let $y = x^k$ and write

$$p(x) = a_n x^n + \cdots + a_1 x + a_0$$

as $$p(x) = p_m(x)y^m + \cdots + p_1(x)y + p_0(x)$$

where $\deg(p_i) \leq k - 1 \quad i = 0,1,\ldots,m$

and $m = \lceil n/k \rceil - 1$.

In fact $$p_0(x) = a_{k-1}x^{k-1} + \cdots + a_1 x + a_0$$

$$p_1(x) = a_{2k-1}x^{k-1} + \cdots + a_{k+1}x + a_k$$

and so on.

Each of these $p_i(x)$ can be computed using only scalar multiplications on $x, x^2, x^3, \ldots, x^{k-1}$.

Therefore, p(x) can be evaluated in m additional non-scalar multiplications as

$$p(x) = (((\cdots(p_m(x)y + p_{m-1}(x))y + p_{m-2}(x))y + \cdots + p_1(x))y + p_0(x).$$

Minimizing $k + n/k - 2$ with respect to k gives $k = \sqrt{n}$

or $2\sqrt{n} - 2$ total non-scalar multiplications required. ___/

Note that Algorithm D uses n additions and about $n - \sqrt{n}$ scalar multiplications. The $- \sqrt{n}$ appears because $\sqrt{n}$ scalars enter through additions.

Algorithm D actually gives a method for producing $O(\sqrt{n})$ non-scalar multiplication algorithms from O(n) algorithms in which all multiplications are counted. The idea is to compute $x^2, x^3, x^4, \cdots, x^k$ for some k, let $y = x^k$, $m = n/k$, and write p(x) of degree n as

$$p(x) = p_m(x)y^m + \cdots + p_1(x)y + p_0(x)$$

$$\text{where } \deg(p_i) \leq k - 1 \qquad i = 0,1,\cdots,m.$$

This polynomial can be evaluated in $O(m) = O(n/k)$ additional non-scalar multiplications using one of the O(n) algorithms in which all multiplications are counted. Using this method on Horner's rule yields Algorithm D. Complications arise if the O(n) algorithm uses preprocessing because the preprocessing must be done before the "coefficients" $p_i(x)$ are evaluated. This method does give insights toward producing better $O(\sqrt{n})$ algorithms. The first of these was obtained from Algorithm C.

<u>Theorem 3.6</u> Any polynomial of degree n can be evaluated using $\sqrt{2n} + O(\log n)$ non-scalar multiplications and no divisions. Moreover, the scalars used by the algorithm are rational functions of the coefficients of the polynomial.

Proof :

Algorithm E. Assume $n = k(2^j - 1)$ for some integers k, j.

1). Compute $x^2, x^3, x^4, \ldots, x^k$ (k multiplications)

2). Compute $x^{2k}, x^{4k}, x^{8k}, \ldots, x^{2^{j-1}k}$ (log n/k multiplications)

Let p(x) be monic of degree $k(2m - 1)$ expressed in the form $p(x) = q(x) \cdot x^{km} + r(x)$

where q is monic, $\deg(q) = k(m-1)$, $\deg(r) \leq km - 1$.

Formally divide $r(x) - x^{k(m-1)}$ by $q(x)$

obtaining $r(x) - x^{k(m-1)} = c(x) \cdot q(x) + s(x)$

where $\deg(c) \leq k - 1$, $\deg(s) \leq k(m - 1) - 1$.

Therefore $p(x) = (x^{km} + c(x))q(x) + (x^{k(m-1)} + s(x))$

or $p(x) = (x^{km} + c(x))q(x) + \hat{s}(x)$

where $\deg(c) \leq k - 1$, $\deg(q) = \deg(\hat{s}) = k(m - 1)$, q and $\hat{s}$ are monic, and the coefficients of c, q, $\hat{s}$ are rational functions of the coefficients of p.

If $m = 2^{i-1}$ for some i, $2 \leq i \leq j$, then p is of the form monic of degree $k(2^i - 1)$, and q and $\hat{s}$ are of the same form monic of degree $k(2^{i-1} - 1)$. Also x^{km} is one of the powers computed in 2). Also $\deg(c) \leq k - 1$ implies that c can be computed free of non-scalar multiplications.

Let M(n) = the number of non-scalar multiplications required to evaluate a degree n monic polynomial assuming that the powers in 1) and 2) above have been computed.

Then by the above argument

$$M(k(2^i - 1)) = 2M(k(2^{i-1} - 1)) + 1 \qquad i = 2,3,\cdots,j.$$

Also, $M(k) = 0$ because it only involves scalar multiplications and additions.

This recurrence relation solves as

$$M(k(2^i - 1)) = 2^{i-1} - i \qquad i = 1,2,3,\cdots,j$$

$$\text{or} \quad M(n) = 2^{j-1} - 1 \approx n/2k .$$

The total number of non-scalar multiplications is

$$n/2k + k + \log n/k .$$

Minimizing with respect to k gives $k \approx \sqrt{n/2}$

or $\sqrt{2n} + \log \sqrt{2n}$ non-scalar multiplications required.

As in Algorithm C, for general n this may required an extra $\log \sqrt{2n}$ multiplications. ___/

Algorithm E uses about $n - \sqrt{2n}$ scalar multiplications.

The number of additions can be counted as

$$A(k(2^i - 1)) = 2A(k(2^{i-1} - 1)) + k + 1$$

$$A(k) = k$$

which solves as

$$A(k(2^i - 1)) = k(2^i - 1) + 2^{i-1} - 1 \qquad i = 1,2,3,\cdots$$

or $A(n) \approx n + n/2k = n + \sqrt{n/2}.$

The next result is obtained from Algorithm B.

<u>Theorem 3.7</u> Any polynomial of degree n can be evaluated using $\lceil \sqrt{2n} \rceil + 2$ non-scalar multiplications and no divisions.

Proof :

Algorithm F. Refer to Algorithm B which evaluates p(x) of degree n as $y = x + c$ $w = y^2$

$$z = (a_n y + s_0)y + t_0 \qquad (n \text{ even})$$

$$z = a_n y + t_0 \qquad (n \text{ odd})$$

$$p(x) = (((\cdots(z(w - s_1) + t_1)(w - s_2) + t_2)\cdots)(w - s_m)$$

$$m = \lceil n/2 \rceil - 1$$

where $c, s_i, t_i \in R \quad i = 0,1,\cdots,m.$

Also compute $w^2, w^3, w^4, \cdots, w^k$ for some k.

Note that if z(x) is any polynomial, then

$$(((\cdots(z(x)(w - s_1) + t_1)(w - s_2) + t_2)\cdots)(w - s_k) + t_k$$

can be written as

$$z(x)q(w) + r(w)$$

where $\deg(q) = k$ and $\deg(r) \leq k - 1$.

In fact $q(w) = (w - s_1)(w - s_2)\cdots(w - s_k)$

and $r(w) = ((\cdots(t_1(w - s_2) + t_2)(w - s_3) + t_3) \cdots)(w - s_k) + t_k$.

Applying this $m/k = n/2k$ times to the form for p(x) above,

$$p(x) = (((\cdots(z \cdot q_1(w) + r_1(w))q_2(w) + r_2(w))\cdots)q_{m/k}(w) + r_{m/k}(w) .$$

where $\deg(q_i) = k, \quad \deg(r_i) \leq k - 1 \qquad i = 1,2,\cdots,m/k.$

Each of the q_i and r_i can be computed using only scalar multiplications and additions on $w, w^2, w^3, \cdots, w^k$.

Therefore, at most $k + n/2k + 2$ non-scalar multiplications are required.

Minimizing with respect to k gives $k = \sqrt{n/2}$

or $\sqrt{2n} + 2$ non-scalar multiplications required. ___/

Note that Algorithm F uses about n additions and about $n - \sqrt{2n}$ scalar multiplications.

Algorithm F uses general algebraic (root-finding) preprocessing just as Algorithm B does.

3.2.2 Comparison of the Algorithms

It is interesting to compare the efficiency of some of these algorithms when evaluating a polynomial of degree n at an $m \times m$ matrix argument.

Let Cost of non-scalar multiplication = m^3

Cost of scalar multiplication = m^2

Cost of addition = m^2

Then Horner's rule has cost $nm^3 + nm^2$

Algorithm D has cost about $2\sqrt{n}\, m^3 + 2nm^2$

Algorithm D is therefore more efficient than Horner's rule if

$$n > 4(m/(m-1))^2$$

Algorithm E may also be compared. Preprocessing algorithms are usually efficient only if the polynomial is to be evaluated enough times to recover the time lost doing preprocessing. However in this case, since the preprocessing involves only operations on numbers (not matricies), preprocessing methods would be more efficient even for a single

evaluation of a polynomial with matrix argument if the polynomial is small enough and the matrix large enough. Since the preprocessing done by Algorithm E is rational, the number of operations involved can be counted.

Let $P(n)$ = the number of preprocessing operations required for degree n polynomial.

The preprocessing for degree $k(2^i - 1)$ is reduced to the preprocessing for two degree $k(2^{i-1} - 1)$ polynomials plus the division of a degree $k \cdot 2^{i-1} - 1$ polynomial by a degree $k \cdot 2^{i-1} - k$ polynomial. This division can be done in about $2k(k(2^{i-1} - 1)) \leq k^2 \cdot 2^i$ operations.

Therefore $P(k(2^i - 1)) \leq 2P(k(2^{i-1} - 1)) + 2^i \cdot k^2$

and $P(k) = 0$,

which solves as $P(k(2^i - 1)) \leq k^2(i - 1) \cdot 2^i$

or $P(n) \leq (\sqrt{n/2})^2 (\log \sqrt{2n})(\sqrt{2n}) \leq (n^{3/2} \cdot \log n)/\sqrt{2}$.

Algorithm E is more efficient than Algorithm D if

$$\sqrt{2n}\, m^3 + 2nm^2 + (n^{3/2} \cdot \log n)/\sqrt{2} < 2\sqrt{n}\, m^3 + 2nm^2$$

or $$n \log n < \approx m^3$$

3.2.3 Conditioning

The reader should be warned that the fact that an algorithm looks efficient on paper does not imply that it is well suited for practical use in all cases. Actual machines are not perfect computing devices but represent numbers only to some fixed decimal accuracy. One numerical

problem that arises concerning polynomial evaluation algorithms is the problem of the conditioning of an algorithmic form. A polynomial evaluation method can be viewed as a transformation which maps a set of parameters having at least $n + 1$ degrees of freedom into the set of all real polynomials of degree n. An algorithm which evaluates a specific polynomial is obtained by substituting the proper numbers for the parameters. For example, in Algorithm B, c, s_1, t_1 are the parameters.

Informally, an algorithmic form is said to be <u>ill-conditioned</u> if "small" errors in the parameters produce "large" errors in the polynomial being evaluated. Ill-conditioning is a property of the transformation defined by the algorithmic form and is independent of round-off effects and other problems which arise during the actual execution of the algorithm. Informally, an algorithm which evaluates a specific polynomial is ill-conditioned if the error in computed values of the polynomial is larger than one would expect from normal round-off effects.

Experiments done by Rice [12] indicate that Algorithm B is more likely to be ill-conditioned than Horner's rule. To my knowledge, no analysis, either experimental or theoretical, has been done concerning the conditioning of the rational preprocessing Algorithm C.

Since the $O(\sqrt{n})$ algorithms D, E, F have basically the

same form as the O(n) algorithms from which they are derived, it could be expected that they would have similar conditioning as algorithms A, C, B, respectively.

For a further discussion of conditioning, see for example [12], [13].

3.3 Possible Improvements

The previous two sections have shown that any polynomial of degree n with algebraically independent coefficients requires at least $\sqrt{n} - 1$ but not more than $\sqrt{2n} + 2$ non-scalar multiplication/divisions for its evaluation. Presumably it should be possible to improve one or both of these bounds to reach some optimal point inbetween.

One way in which the lower bound might be raised is as follows. Notice that the algorithmic scheme A_k (page 23) actually generates a result of degree 2^k if it contains no divisions.

$$p(x) = p_{2^k}(\vec{m})x^{2^k} + \cdots + p_1(\vec{m})x + p_0(\vec{m})$$
$$p_i \in Q[\vec{m}] \qquad i = 0,1,2,\cdots,2^k.$$

If A_k is computing a polynomial of degree n,

$$p(x) = a_n x^n + \cdots + a_1 x + a_0$$

then not only $p_i(\vec{m}) = a_i \quad i = 0, 1, 2, \cdots, n$

but also $p_i(\vec{m}) = 0 \quad i = n+1, n+2, \cdots, 2^k$

The equations $p_i(\vec{m}) = 0 \quad i = n+1, n+2, \cdots, 2^k$ place constraints on the parameters $\vec{m}$ so that they cannot have

their full $k^2 + 4k + 2$ degrees of freedom but something less. If A_k contains divisions, then it is possible to keep the degree of the result small, but again some degrees of freedom in the parameters must be lost to assure that A_k is computing a polynomial and not a general rational function.

Algorithm F ($\sqrt{2n}$ non-scalar multiplications) is probably not optimal because the first $k = \sqrt{n/2}$ multiplications $w^2, w^3, \ldots, w^k$ are wasted in the sense that they introduce no parameters into the algorithm. There probably are schemes which use fewer than $\sqrt{2n}$ non-scalar multiplication/divisions but require complicated preprocessing. Given $a_0, a_1, \cdots, a_n$, the preprocessing would require the solution of $p_i(\vec{m}) = a_i \qquad i = 0,1,2,\ldots,n$ for the parameters $\vec{m}$, where the p_i are complicated rational functions in $\vec{m}$.

Thus $\sqrt{n}$ appears to be the best lower bound obtainable by a simple degrees of freedom argument alone. $\sqrt{2n}$ is probably close to the best algorithm with a simple description and reasonable preprocessing.

Another gap presently exists concerning the number of multiplications required to evaluate a polynomial with algebraically independent coefficients and with rational preprocessing. The best known algorithm (Algorithm C) uses $n/2 + O(\log n)$ multiplications. Any improvement of the

n/2 lower bound (Theorem 2.2) assuming rational preprocessing will require an argument which goes beyond simple degrees of freedom arguments. Of course, there may be a better algorithm also.

3.4 Lower Bounds on Efficient Evaluation

As shown in section 3.2, any rational polynomial can be evaluated in $O(\sqrt{n})$ multiplications. This economy of multiplications is gained at the expense of doing a very large number of additions. Since addition is never actually free, this method must be inefficient in general. This section will show that almost all rational polynomials of degree n require n/2 multiplication/divisions for their efficient evaluation if addition has positive cost.

Let $c_a \in R$, $c_a \geq 0$, denote the cost of addition and $c_m \in R$, $c_m \geq 0$, denote the cost of multiplication.

Define the cost of a rational algorithm to be

$$n_a \cdot c_a + n_m \cdot c_m$$

where n_a = number of addition/subtractions used

n_m = number of multiplication/divisions used.

Theorem 3.8 If $c_a > 0$, then almost all rational polynomials of degree n require at least $\lceil n/2 \rceil$ multiplication/divisions for their evaluation by any least cost rational algorithm over R.

Proof : Assume $q(x) = q_n x^n + \cdots + q_1 x + q_0 \in Q[x]$ can be evaluated in no more than k multiplication/divisions and therefore by the scheme S_k (see page 12).

Assume $k < \lceil n/2 \rceil$.

$q(x)$ can be evaluated by Horner's rule with cost $n(c_a + c_m)$. Let $B = \lceil n(c_a + c_m)/c_a \rceil$.

Any algorithm which evaluates $q(x)$ using more than B addition/subtractions cannot be a least cost algorithm.

Consider each of the algorithmic forms obtained from S_k by independently choosing a $*$ or $\div$ at each step u_r, and choosing some substitution of integers for the m_{ij}, $\hat{m}_{ij}$ such that the algorithmic form uses $\leq$ B addition/subtractions. Remember that $m_{ij}u_j$ is shorthand for m_{ij} additions. There are only a finite number N of such forms.

The i-th form computes a result of the form

$$p(x) = p_{in}(\vec{s})x^n + \cdots + p_{i1}(\vec{s})x + p_{i0}(\vec{s})$$
$$= q_n x^n + \cdots + q_1 x + q_0$$

where $\vec{s}$ is of length $< n + 1$.

Therefore, for $i = 1, 2, \cdots, N$, there is a

$P_i \in Q[y_n, \cdots, y_0]$, $P_i \not\equiv 0$, such that

if $q_n x^n + \cdots + q_1 x + q_0$ is computed by the i-th form

then $P_i(q_n, \cdots, q_0) = 0$.

Define $P(y_n, \cdots, y_0) = \prod_{i=1}^{N} P_i(y_n, \cdots, y_0)$.

Then $q_n x^n + \cdots + q_1 x + q_0$ computed by a least cost algorithm using less than $\lceil n/2 \rceil$ multiplication/divisions

implies that $P(q_n, \cdots, q_0) = 0$. ___/

As the ratio c_a/c_m grows smaller, there are more rational polynomials that can be evaluated using less than n/2 multiplication/divisions, but as long as $c_a/c_m > 0$, almost all require n/2.

A similar argument shows that almost all rational polynomials of degree n require n additions for their evaluation by an efficient algorithm.

Theorem 3.9 If $c_a > 0$ and $c_m > 0$, then almost all rational polynomials of degree n require at least $\lceil n/2 \rceil$ multiplication/divisions and at least n addition/subtractions for their evaluation by any least cost rational algorithm over R.

Proof : We use the fact that for every rational algorithmic form using some number of multiplication/divisions and less than n addition/subtractions, there is a

$$P_i \in Q[y_n, \cdots, y_0], \quad P_i \not\equiv 0, \quad \text{such that}$$

if $q_n x^n + \cdots + q_1 x + q_0$ is computed by that form,

$$\text{then } P_i(q_n, \cdots, q_0) = 0.$$

This follows from the fact that if the form contains less than n addition/subtractions, then the coefficients of the result of the form can be parameterized in less than $n + 1$ parameters. (See Theorem 2.3 and Knuth [4]).

The number of multiplication/divisions in a least

cost algorithm can be bounded by $\hat{B} = \lceil n(c_a + c_m)/c_m \rceil$

There are only a finite number of algorithmic forms using fewer than n addition/subtractions and not more than $\hat{B}$ multiplication/divisions. As in Theorem 3.8, construct a $\hat{P} \in Q[y_n, \cdots, y_0]$, $\hat{P} \not\equiv 0$, such that if $q_n x^n + \cdots + q_1 x + q_0$ is computed by a least cost algorithm using less than n addition/subtractions,

then $\hat{P}(q_n, \cdots, q_0) = 0$.

Let $P(y_n, \cdots, y_0)$ be as in Theorem 3.8.

Define $\bar{P}(y_n, \cdots, y_0) = P(y_n, \cdots, y_0) \cdot \hat{P}(y_n, \cdots, y_0)$.

The conclusion follows. ___/

Theorem 3.9 states that any computational savings that take advantage of the algebraic dependence of rational numbers can never work for all rational polynomials.

The last two lower bound results allow preprocessing. It is reasonable to ask what kind of lower bound for rational polynomial evaluation can be proven assuming no preprocessing. The question is almost answered by Theorem 2.5 which states that any algorithm which computes the general polynomial

$$P(x, a_0, a_1, \cdots, a_n) = a_n x^n + \cdots + a_1 x + a_0$$

requires at least n multiplication/divisions.

Notice, however, that there are no-preprocessing algorithms that use less than n multiplication/divisions and work for some infinite set of real polynomials.

For example

$$\hat{P}(x, a_0, a_1, \cdots, a_n) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

can be evaluated in $n - 1$ multiplications and the algorithm works for all monic polynomials but gives the wrong answer for any non-monic polynomial. Therefore, a no-preprocessing algorithm which used less than n multiplication/divisions, gave wrong answers in general for all real polynomials, but happened to give the right answers for all rational polynomials would not directly contradict Theorem 2.5. However, this cannot happen because the rationals are dense in the reals.

Theorem 3.10 Any rational algorithm over R which evaluates $P(x, a_0, a_1, \cdots, a_n) = a_n x^n + \cdots + a_1 x + a_0$ correctly for all $a_0, a_1, \cdots, a_n \in Q$ requires at least n multiplication/divisions.

Proof : Formally carrying out the operations in the no-preprocessing algorithm, the algorithm is seen to compute a result of the form

$$p(x) = p_n(\vec{a})x^n + \cdots + p_1(\vec{a})x + p_0(\vec{a})$$

where $\vec{a} = (a_0, \cdots, a_n)$

and $p_i \in R(a_0, \cdots, a_n) \qquad i = 0,1,\cdots,n$,

that is, the p_i are rational functions in the a_i with real coefficients.

Let $\vec{t} \in R^{n+1}$ be any real vector such that

$$p_i(\vec{t}) \neq \infty \qquad i = 0,1,\cdots,n.$$

Pick a region about $\vec{t}$ within which all the p_i are continuous.

Within this regionm find a sequence of rational vectors $\vec{q}_i \in Q^{n+1}$ $i = 1,2,3,\cdots$ such that $\lim \vec{q}_i = \vec{t}$.

Since the algorithm works for all rational polynomials, we have $(p_0(\vec{q}_i), \cdots, p_n(\vec{q}_i)) = \vec{q}_i$ $i = 1,2,3,\cdots$

Therefore

$$\begin{aligned}(p_0(\vec{t}), \cdots, p_n(\vec{t})) &= (p_0(\lim \vec{q}_i), \cdots, p_n(\lim \vec{q}_i)) \\ &= \lim (p_0(\vec{q}_i), \cdots, p_n(\vec{q}_i)) \\ &= \lim \vec{q}_i = \vec{t}\end{aligned}$$

Therefore the algorithm works for all real polynomials for which it does not diverge. (It can diverge only on a set of measure zero). Therefore, by Theorem 2.5, the algorithm requires n multiplication/divisions. ___/

Algorithm D was stated to be a no-preprocessing algorithm which could be used to evaluate any rational polynomial in $2\sqrt{n}$ multiplications. If this algorithm is used for rational polynomial evaluation, there actually is implicit preprocessing because $z \cdot u$, $z \in Z$, must be translated into $u + u + \cdots + u$ (z times). There is no way for a rational algorithm to do this. However, an algorithm with comparison and branching instructions could.

It is not hard to construct an algorithm using the perfect real arithmetic operations and comparison and branching instructions which would evaluate

$$P(x, a_0, a_1, \cdots, a_n) = a_n x^n + \cdots + a_1 x + a_0$$

correctly for all $a_0, a_1, \cdots, a_n \in Q$, $x \in R$

using $2\sqrt{n}$ multiplications and no divisions (and many additions and comparisons), but would diverge if any of the a_i were irrational. We assume that the a_i and x are initially in some n+2 input registers, each of which is capable of holding a perfect real number.

Therefore, the assumption of rational algorithm is necessary for a result such as Theorem 3.10.

IV. Summary.

The main purpose of this work has been to exhibit bounds on the number of arithmetic operations required to evaluate rational polynomials.

Since integer multiplications can be done by successive additions, the analysis of lower bounds was facilitated by only counting multiplication/divisions that involve x, the argument variable of the polynomial being evaluated, on both sides. These are called non-scalar multiplication/divisions. The first lower bound result (Theorem 3.4) shows that almost all rational polynomials of degree n require $\sqrt{n}$ (non-scalar) multiplication/divisions for their evaluation. Almost all rational polynomials are all those off some set of measure zero.

Several algorithms were presented that can evaluate any polynomial of degree n using $O(\sqrt{n})$ non-scalar multiplications, O(n) scalar multiplications, and O(n) additions. These algorithms can in theory be used to evaluate any rational polynomial of degree n using $O(\sqrt{n})$ total multiplications by doing integer scalar multiplications by many additions. These algorithms may have practical use in cases where non-scalar multiplications are more expensive than scalar multiplications. This would be true if the polynomial is being evaluated at a matrix argument or at some other mathematical object for which it is harder to multiply two

objects than it is to multiply an object by a number.

The trick of simulating an integer multiplication by many additions must give inefficient algorithms in general. This is formalized by a lower bound result (Theorem 3.9) which shows that almost all rational polynomials of degree n require n/2 multiplication/divisions and n addition/subtractions for their efficient evaluation. These are the same (achievable) lower bounds that apply to any polynomial with algebraically independent coefficients.

The techniques described herein could easily be applied to extend other degrees of freedom arguments to apply to rational numbers. Consider the following theorem of Winograd.

Theorem (Winograd [6]). If A is a $p \times q$ matrix whose entries form a set of algebraically independent real numbers and if $\vec{v}$ is a q-vector input, then any rational algorithm which computes the p elements of $A \cdot \vec{v}$ requires at least $\lceil \frac{1}{2}pq \rceil$ multiplication/divisions.

Using techniques of Theorem 3.9, this can be extended to the following.

Theorem If addition has positive cost, then for almost all $p \times q$ matricies A with rational entries, any least cost rational algorithm which computes $A \cdot \vec{v}$ requires at least $\lceil \frac{1}{2}pq \rceil$ multiplication/divisions.

REFERENCES

1. T. S. Motzkin, "Evaluation of Polynomials and Evaluation of Rational Functions", Bulletin American Math. Soc. 61(1955), p. 163.

2. V. Ya Pan, "Methods of Computing Values of Polynomials", Russian Math. Surveys 21(1966), pp. 105-136.

3. J. Eve, Numerische Mathematik 6(1964), pp. 17-21.

4. D. E. Knuth, "Seminumerical Algorithms", Vol. 2 of "The Art of Computer Programming", Addison-Wesley, Reading, Mass., pp. 422-444.

5. B. L. van der Waerden, "Modern Algebra", Ungar, New York, 1949.

6. S. Winograd, "On the Number of Multiplications Required to Compute Certain Functions", Communications on Pure and Applied Math 23(1970), pp. 165-179.

7. E. G. Belaga, Problemi Kibernetiki 5(1961), pp. 7-15.

8. A. Borodin, "Some Results on the Computation of General Polynomials", University of Toronto, unpublished manuscript.

9. G. Estrin, "Organization of Computer Systems - The Fixed Plus Variable Structure Computer", Proceedings Western Joint Computer Conference (May, 1960), pp. 33-40.

10. W. S. Dorn, "Generalizations of Horner's Rule for Polynomial Evaluation", IBM Journal of Research and Development, 6(1962), pp. 239-245.

11. I. Munro and M. Paterson, "Optimal Algorithms for Parallel Polynomial Evaluation", Proceedings of 12th Annual Symposium on Switching and Automata Theory, East Lansing, Mich., 1971, pp. 132-139.

12. J. R. Rice, "On the Conditioning of Polynomial and Rational Forms", Numer. Math. 7(1965), pp. 426-435.

13. J. R. Rice, "A Theory of Condition", J. SIAM Numer. Anal. 3(1966), pp. 287-310.