Department of Applied Analysis
and Computer Science
Research Report CSTR 1013
April, 1972

EFFICIENT EVALUATION OF POLYNOMIAL FORMS

by
Ian Munro, University of Waterloo
and
Allan Borodin, University of Toronto

Faculty of Mathematics
University of Waterloo
Waterloo, Ontario
Canada

Department of Applied Analysis
&
Computer Science

Department of Applied Analysis
and Computer Science
Research Report CSTR 1013
April, 1972

EFFICIENT EVALUATION OF POLYNOMIAL FORMS

by
Ian Munro, University of Waterloo
and
Allan Borodin, University of Toronto

ABSTRACT

The evaluation of several polynomial forms is considered. New algorithms for the evaluation of a polynomial and its derivative, a polynomial at two points, a polynomial of high degree using multiple precision arithmetic, and a bivariate polynomial of the form $\Sigma\, a(i)\, x^{i} y^{n-i}$ are presented. Various "coefficient splitting techniques" are introduced in these algorithms and the optimality of certain techniques is shown.

Some of the results described have been presented in preliminary form in [11] and [12].

Introduction

Computational Complexity is concerned with how difficult, under some measure of difficulty, it is to evaluate certain functions or classes of functions. Historically, much of the work in this area deals with models of computation quite unlike a stored program computer as it is commonly envisioned. Furthermore, the functions dealt with are typically very different from those which are commonly computed. The reason for this is quite simply that researchers are attacking the deep problem of what makes a function hard to compute. The general approach has been, given a model of computation, to consider first a complexity bound for the computation, and then to show that certain types of functions are, or are not, computable within this bound.

More recently, there has been a trend to consider first the function or class of functions and then to ask how much "work" is required to perform the evaluation. Typically these functions are of a type computed in practice, the model of computation is an idealization of a digital computer, and the measure of complexity is the number of operations of a certain type which are performed. There are two basic immediate goals of such studies. The first, and most obvious, is to develop "good" algorithms for computing functions which frequently occur in computational practice. The second, and generally much harder goal is to show that the evaluation of certain classes of functions require so many operations of a certain type. Hopefully these bounds will demonstrate that the algorithms developed are optimal for use on most digital computers. The ultimate goal is, of course, the same as that of the other branches of Computational Complexity, to understand what makes certain functions hard to compute.

In this paper we present some results concerning the evaluation of certain types of polynomials and evaluation of powers of numbers. We present a technique of splitting up the coefficients of a polynomial which permits faster computation of several polynomial forms. We also extend a technique for proving lower bounds on the number of multiplications and divisions required for computations, to prove that the number of these operations needed to compute certain polynomial forms is somewhat greater than the number of inputs to the problem.

The model of computation envisioned in this paper is a random access register machine in which each register may store an arbitrary integer or floating point number. The machine has four basic operations, addition, subtraction, multiplication and division. The functions with which we shall be concerned are calculable with a finite predetermined number of arithmetic operations and hence we shall dispense with looping and branching instructions in the formal model. Algorithms will, however, be written in a form using looping, and at times recursion, in order to conceptualize them more clearly. The basic aim is, to present an algorithm and then show that it uses the minimum number possible of each type of operation in computing the function. Most existing proof techniques, including the ones used here, show that at least so many operations of a particular form must be used. This form may be very restrictive. If an algorithm achieves this bound and uses no other types of operations, the result attained is very strong. If the algorithm minimizes one type of operation at the expense of another, the shortcomings of the proof techniques become apparent.

In this paper, attention is given to reducing the number of multiplications and divisions required. We do not justify this by falsely claiming that single precision multiplications take much longer than additions. Rather, minimizing the number of multiplications and divisions becomes important if the numbers concerned are very large integers, real numbers stored to high precision, or the computations are symbolic. Similar emphasis is given in [3], [7], [11], [12], [16], [17], [18], and [19]. In particular, Strassen [17] has announced independent discovery of Theorems 1 and 4. This approach is further justified by the observation that by minimizing the number of multiplications and divisions required for a computation, the total number of arithmetics is often reduced as well. However, the problem of minimizing additions and subtractions, and more important, the total number of arithmetics, should not be neglected. Kirkpatrick [9] and others have developed techniques for proving lower bounds on the number of additions and subtractions needed. There are, unfortunately few techniques for dealing with the problems of tradeoffs between various types of operations.

The Evaluation of a Polynomial and its First Derivative

One computation which has received a good deal of attention is the efficient evaluation of the general polynomial of degree n. In proving lower bounds on the number of multiplications and divisions required, the definition of certain types of these operations as active or inactive is very useful.

A multiplication or division f{op}g is inactive if one or more of the following hold:

(1) g is a constant
(2) f is a constant and the operation is multiplication
(3) Neither f nor g depend on the coefficients of the polynomial
(4) The operation is equivalent to one which satisfies one of the above conditions.

A multiplication or division is said to be an active operation if it is not inactive. Hence in the evaluation of $P(\vec{a},x) = \sum_{i=0}^{n} a(i)x^{i}$, $a(i) \cdot 2$ and $x \cdot x$ would be inactive, but $a(i) \cdot x$ is an active multiplication. Suppose we first compute $P_1 = a(1) \cdot a(1)-1$, then condition (4) implies that the division $P_2=P_1/(a(1)+1)$ is inactive. Pan [13] has shown that the evaluation of a general polynomial of degree n requires at least n active mult/div. A more direct proof is given by Hopcroft and Borodin [2]. Hence, the well-known "Horner's rule" minimizes the number of multiplications and divisions for evaluating a polynomial from its coefficients, even if all the powers of the indeterminate happen to have been calculated previously. Belaga [1] has shown that Horner's rule also minimizes additions and subtractions. From Pan's result we have a direct proof of the following theorem due to Winograd [18]. A similar result may also be attained for additions and subtractions.

Theorem 1 - Given their coefficients, the evaluation of m unrelated polynomials of degrees $n_1, \ldots, n_m$ at the same point in general requires at least $\sum_{i=1}^{m} n_i$ active operations; and so an optimal method for their evaluation is the repeated application of Horner's rule.

Proof - Let the i^{th} polynomial, $P_i(x)$, be defined as

$$P_i(x) = \sum_{j=0}^{n_i} a(i,j)x^j$$

Then consider the general polynomial of degree $\sum_{i=1}^{m} (n_i+1)-1$ given by

$$P(x) = \sum_{i=1}^{m} P_i(x) \cdot x^{s(i)} \quad \text{where} \quad s(i) = \sum_{j=1}^{i-1} (n_j+1)-1$$

The evaluation of this polynomial requires at least $(\sum_{i=1}^{m} n_i)+m-1$ mult/div even if we do not count those used to evaluate powers of x. Suppose we have a method of evaluating all the $P_i(x)$'s in N active multiplications. Then the definition of P gives an algorithm for its calculation in N+m-1 multiplications not counting evaluation powers of x.

Hence $$N+m-1 \geq \sum_{i=1}^{m} n_i+m-1$$

so $$N \geq \sum_{i=1}^{m} n_i$$

Q.E.D.

The evaluation of both a polynomial and its first derivative is slightly different, as the concept of active multiplications is not adequate for determining the minimum number of multiplications and divisions in which this computation may be performed.

Theorem 2 - The evaluation of a polynomial of degree n and its first derivative requires at least n active mult/div and can be performed within this bound. However, for $n \geq 3$ a total of more than n mult/div is required.

Proof - By the previously mentioned theorem of Pan, at least n active mult/div are necessary to evaluate a polynomial and its derivative. The following algorithm computes the polynomial $\text{poly} = \sum_{i=0}^{n} a(i)x^i$ and its derivative (deriv) in 2n-2 multiplications, but only n active multiplications, when $n \geq 1$. The algorithm is intended only to illustrate this point.

Algorithm 1

```
    poly   ←  0
    deriv  ←  0
    x(i)   ←  $x^i$      i = 1, ..., n-1
[this step requires n-2 inactive multiplications]
    for  i = 1, ..., n-1
    begin
         poly  ←  poly  +  a(n-i+1)  ×  x(n-i)
         deriv ←  deriv +  poly
    end   [n-1 active multiplications are used in this loop]
    poly   ←  poly  +  a(1)
    deriv  ←  deriv +  poly
    poly   ←  poly  ×  x   + a(0) .
```

It has been shown by Borodin [2] that Horner's rule is an essentially unique method of polynomial evaluation in n multiplications. Hence for $n > 2$, more than n multiplications are needed to evaluate both the polynomial and its derivative, and so more than an active multiplication argument is needed to determine a lower bound for the complexity of this computation.

Q.E.D.

Hopcroft [8] has suggested a $[1+\varepsilon]n$ algorithm. A procedure is given which requires $n+2\sqrt{n}$ multiplications and $2n+2\sqrt{n}$ additions. This economy of total arithmetics is at the expense of creating an array of length approximately $\sqrt{n}$. The evaluation of $\sqrt{n}$ need only be a rough approximation. This $\sqrt{n}$ splitting of a polynomial will be used to compute several other polynomial forms using fewer than the number of multiplications (or operations) of "conventional" methods.

Algorithm 2

```
poly   ←  0
deriv  ←  0
m      ←  $\lfloor sqrt(n-1) \rfloor$
m2     ←  $\lfloor n/m \rfloor$
x(i)   ←  $x^i$        (i = 0, ..., m)
                   [m-1 multiplications]

k  ←  n
for   i = 1, ..., m2
begin
    poly   ←  (poly+a(k))×x(m)
    k      ←  k-1
    deriv  ←  deriv  ×  x(m)  +  poly
    for  j = m-1, ..., 1
```

```
        begin
            poly   ←  poly  +  x(j)  ×  a(k)
            k      ←  k-1
            deriv  ←  deriv  +  poly
        end [m-1 multiplications in this loop]
    end     [m2 · (m+1) multiplications in this loop]
    for     k = n-m2×m, ..., 1
    begin
        poly  ←  (poly+a(k))  ×  x(k-1)
        deriv ←  deriv  +  poly
    end  [n-m2 · m multiplications in this loop]
    poly  ←  poly  ×  x  +  a(0)
    [a total of n+m+m2 multiplications are needed for this procedure]
```

The algorithm above is essentially the same as one independently developed by Paterson and Stockmeyer [14] for the evaluation of a polynomial in which the coefficients are real, but the indeterminate is a very large matrix. The operation of major concern in their case is the matrix by matrix multiplication. Note that, if all computations involving "deriv" are ignored, Algorithm 2 computes "poly" using about $2\sqrt{n}$ multiplications in which both multiplicands depend on x.

Evaluation of a Polynomial at More than One Point

Very often the same polynomial is to be evaluated at several points. Such computations fall into two basic categories. To use a bit of Turing machine terminology, they are the on-line and off-line models of computation. The on-line case is usually a situation in which the polynomial is evaluated at one point and on the basis of this value the next point is determined. This is typically the case in problems such as finding roots of polynomials. In such cases, the value of the polynomial at the ith point must be determined before the i+1st point is given. The off-line case is simply the situation in which all, or at least a good number, of points are given at once. These distinctions may seem minor, however, the evaluation is asymptotically (as the degree of the polynomial and number of points become large) much faster in the off-line case.

Pan [13] and others have developed schemes by which functions of the coefficients of the polynomial may be computed (once), and these values used to compute the function in $\lfloor n/2 \rfloor+2$ multiplications and n additions. Such techniques are generally referred to as preconditioning methods. Motzkin [10] and Belaga [1] have shown that such methods are almost optimal for on-line polynomial evaluations as $\lfloor n/2 \rfloor+1$ multiplications and divisions, and n additions and subtractions are required to evaluate a general polynomial regardless of how much preconditioning occurs.

Such results, of course, do not imply that the evaluation of a polynomial of degree n at m points requires 3mn/2 arithmetics. It is shown in [3] that a $\sqrt{n}$-splitting and fast matrix multiplication may be used to evaluate a polynomial of degree n at $\sqrt{n}$ points in essentially the time required to multiply two $\sqrt{n}$ by $\sqrt{n}$ matrices. Strassen [16] has shown that two k by k matrices may be multiplied in $O(k^{\ell og\ 7})$[1] $\approx O(k^{2.81})$ arithmetics. Others have speculated that the exponent may be very "close to" 2. In any case, if $m \geq \sqrt{n}$, a polynomial of degree n may be evaluated at any m points in $O(mn^{.91})$ or fewer operations. That is, less than linear in n when viewed on a polynomial-point basis.

In attacking the problem of multiplying large matrices quickly, Fiduccia [7] has developed a technique for multiplying a k by k matrix by two vectors in about $3k^2/2$ multiplications rather than $2k^2$. This algorithm, together with the $\sqrt{n}$-splitting technique may be used to evaluate a polynomial at any two points in $3n/2+O(\sqrt{n})$ multiplications. Let $P(x) = \sum_{i=0}^{n} a(i)x^i$ and without loss of generality assume again that n is a perfect square. The technique is essentially that of [3], write

$$A = \begin{bmatrix} a(1) & a(2) & \ldots\ldots & a(\sqrt{n}) \\ a(\sqrt{n}+1) & \ldots\ldots\ldots\ldots\ldots\ldots & & \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & & & a(n) \end{bmatrix}$$

and then, if x_1 and x_2 are the points of evaluation,

$$X = \begin{bmatrix} x_1 & x_2 \\ x_1^2 & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ x_1^{\sqrt{n}} & x_2^{\sqrt{n}} \end{bmatrix}$$

X is found in $2\sqrt{n} - 2$ multiplications. Then $Y = AX = (y_{ij})$ may be found in roughly $3n/2$ more multiplications

[1] all logarithms are to base 2, unless otherwise noted.

and

$$\sum_{i=0}^{n} a(i)x_j^i \quad = \quad a(0) + y_{1j} + \sum_{k=2}^{\sqrt{n}} y_{kj}\, x_j^{(k-1)\sqrt{n}} \qquad j = 1, 2.$$

is computed in $2\sqrt{n} - 2$ more. The entire process takes $3n/2 + O(\sqrt{n})$ multiplications. The point of this algorithm is more to illustrate that the number of multiplications used in such a computation may be reduced from $2n$, rather than to suggest an algorithm for most computational situations, since the number of additions used is such that more than $4n$ (as in two applications of Horner's rule) total arithmetics are required. With the provision that divisions are not used, Kirkpatrick [9] has shown that 2n additions and subtractions are needed for this problem, and so two applications of Horner's rule minimize these operations. We conjecture that a total of 4n arithmetics are needed and that 3n/2 multiplications are required. The best lower bounds presently obtainable are n+1 multiplications/divisions (by uniqueness of Horner's Rule) and hence 3n+1 total arithmetic operations.

Multiple Precision and Symbolic Polynomials

Our final example of the use of a coefficient splitting technique is in the evaluation of a polynomial in which the amount of work required for an arithmetic operation depends on the size or precision of the inputs. In particular we are concerned with the case in which the coefficients and indeterminate are large integers (of almost full word size) and so the product of n of these is an n precision number. That is, the value of the polynomial may be expected to be an $n + 1$ precision integer. Essentially the same problem exists if the inputs (coefficients and indeterminate) are floating point numbers, but the computation must be carried out to full precision. Another version of the same problem is a special case of symbolic polynomial evaluation. Suppose each of the coefficients of the polynomial, and the indeterminate are themselves dense (i.e. most coefficients are non-zero) polynomials of roughly the same degree in the same variable. The function to be computed is the symbolic polynomial in these polynomials. For simplicity, however, we shall consider principally the case in which the inputs are large integers and make only the occasional reference to the other cases.

Using the most obvious method, the product of a k-precision and an ℓ-precision number (or symbolic polynomials of these degrees) requires $O(k \cdot \ell)$ operations. However, using fast Fourier transform-like techniques ([5], [15]) this may be reduced to or almost to $O(r \ell og\ r)$ operations, where $r = \max(k,\ell)$. We exploit these methods to evaluate a polynomial of degree n in a manner asymptotically faster than Horner's rule.

Assume all inputs are single precision integers; if Horner's rule is used, the product of $(\sum_{j=0}^{i} a(n-i+j)x^j)$ and x requires $O(i)$ single precision operations. Hence the entire evaluation requires $O(n^2)$ operations. It can be verified that if multiplication is to be carried out by an $O(k\cdot\ell)$ method that $O(n^2)$ operations are required to evaluate x^n. Hence, under these conditions, Horner's rule is within a constant factor of being optimal. However, if we assume the use of an $O(r\ \ell og\ r)$ multiplication scheme an $O(n\cdot\ell og^2\ n)$ bound is attainable. In fact if m(n), the time required to multiply two n precision numbers, grows reasonably uniformly with n, this bound may be written as $O(m(n)\cdot\ell og\ n)$.

In the following algorithm, which computes poly = $\sum_{i=0}^{n} a(i)x^i$, we assume $a(k) = 0$ for $k > n$.

Algorithm 3

```
xp ← x
for  i = 1, ..., $\lceil \ell og(n+1) \rceil$
begin
    for  j = 0, ..., $\lfloor n/2^i \rfloor$
      a(j) ← a(2j+1) × xp + a(2j)
    xp ← xp × xp
end
poly ← a(0)
```

In each pass through the outer loop of the algorithm the degree of the polynomial to be computed is halved, while the precision of each of the inputs is doubled. On each pass through this loop $\lceil (n+1)/2^i \rceil + 1$ multiplications and $\lceil (n+1)/2^i \rceil$ additions, each operating on two 2^{i-1} precision numbers are performed. The time for the entire process

$$T(n) \leqslant \sum_{i=1}^{\lceil \ell og(n+1)\rceil} (\lceil (n+1)/2^i\rceil+1)(m(2^{i-1})+\alpha(2^{i-1}))$$

where $\alpha(r)$ denotes the time needed to add two r precision numbers. If we assume r-precision multiplication may be performed within $O(r\ \ell og\ r)$ operations. Then $m(r)+\alpha(r) \le O(r\ \ell og\ r)$.

Hence

$$T(n) \le \sum_{i=1}^{\lceil \ell og(n+1)\rceil} (\lceil (n+1)/2^i\rceil+1)\ O(2^{i-1}(i-1))$$

$$= O(n\ \ell og^2\ n).$$

Evaluation of Powers of Numbers

We will now deviate briefly from conventional polynomial evaluation problems to consider the problem of raising numbers to powers.

Consider the problem of evaluating the nth power of a number in the minimum number of multiplications. If the number is real, and multiplication the only operation permitted,the problem is equivalent to generating the number n, starting with 1 and using only additions. The familiar binary algorithm takes at most 2 ℓog n steps. Brauer [4] has demonstrated a method requiring only

$\ell og\ n + 2\ \frac{\ell og\ n}{\ell og(\ell og\ n)}$ steps and Erdös [6] has shown this method to be optimal to within a constant multiple of the lower order term for most choices of n. Briefly, the method is given as follows:

For any b, n may be written uniquely as

$$n = \alpha_1 \cdot 2^b + \beta_1$$

$$\alpha_j = \alpha_{j+1} \cdot 2^b + \beta_{j+1}$$

where $0 \le \beta_j < 2^b$

for $j = 1, \ldots, \frac{1}{b}\ \ell og\ n$

Note that the β_j's are merely b bit sections of the binary representation of n. Thus we may build n by first forming the numbers from 1 to 2^b (that requires 2^b-1 steps). This gives all β_j and also α_s where $s = \frac{1}{b}\ \ell og$ n. The α_j are then formed in descending order. The formation of each requires b+1 steps and this must be done $\frac{1}{b}\ \ell og$ n times to yield n.

Therefore the number of multiplications needed to find x^n is about $2^b + \frac{b+1}{b}\ \ell og$ n for any integer b. This bound may be reduced by judicious choice of b as a function of n. The desired result is attained when $b = \ell og(\ell og\ n) - \ell og(\ell og(\ell og\ n))$.

Looking back at this algorithm we note that most of the steps are doublings, or squarings in the formation of x^n. This fact becomes interesting if the number to be exponentiated is complex, since

$$\begin{aligned}(x+iy)^2 &= x^2-y^2+2xyi\\ &= (x+y)(x-y)+2xyi\end{aligned}$$

and so requires only 2 real multiplications to be computed. A general complex product $(a+bi)(c+di)$ may be found in three real multiplications as

$$\underset{(1)}{a(c+d)}-\underset{(2)}{d(a+b)} = ac-bd$$

$$a(c+d)+\underset{(3)}{d(b-a)} = ad+bc$$

It has been shown independently by Winograd [19] and Munro [11] that this number may not be reduced to 2. Hence if the above exponentiation scheme is used to evaluate a power of a complex number

$$2\ \ell og\ n+6\ \frac{\ell og\ n}{\ell og(\ell og\ n)}$$

multiplications would be needed. It is conjectured that this method is optimal, to within a constant multiple of the lower order term, for most values of n. We note that if addition, subtraction and multiplication are the only operations permitted, then

Theorem 3 - If there exists a k, such that $(x+iy)^k$ can be evaluated in fewer than $2\ \ell og\ k$ multiplications, then $(x+iy)^n$ can be evaluated in fewer than $2\ \ell og\ n$ multiplications for almost all n.

Proof - The technique used to efficiently evaluate the powers of complex numbers is essentially the one previously described. The key difference is that the basic operation is to raise a complex number to the k^{th} power rather than to square it.

Suppose $(x+iy)^k$ requires $2\ \ell og\ k-\varepsilon$ real multiplications to evaluate for some $\varepsilon > 0$ (clearly $2\ \ell og\ k-\varepsilon$ must be an integer, but $2\ \ell og\ k$ need not).

Then for any b, n may be written uniquely as

$$n = \alpha_1 \cdot k^b + \beta_1$$

$$\alpha_j = \alpha_{j+1} \cdot k^b + \beta_{j+1}$$

$$\text{where} \quad 0 \le \beta_j < k^\beta$$

$$\text{for} \quad i = 1, \ldots, \frac{1}{b}\ \ell og_k\ n$$

An algorithm similar to the one previously given may then be used to generate n or rather $(x+iy)^n$. By setting $b = \ell og_k\ \ell og_k\ n - \ell og_k\ \ell og_k\ \ell og_k\ n$ the total number of multiplications required to compute $(x+iy)^n$ will be about

$$3(k^b-1) + \frac{b(2\ \ell og\ k - \varepsilon) + 3}{b}\ \ell og_k\ n$$

$$= 2\ \ell og\ n - \varepsilon\ \ell og_k\ n + \frac{6\ \ell og_k\ n}{\ell og_k(\ell og_k\ n)}$$

Hence for all $n > k^{k^{\frac{6c}{\varepsilon}}}$ (for some c) the evaluation of $(x + iy)^n$ requires fewer than $2\ \ell og\ n$ multiplications.

O.E.D.

This theorem may be of use in attempting to show that there is no k such that $(x+iy)^k$ can be evaluated in fewer than $2 \ell og\ k$ multiplications. Another line of attack on the same problem is to consider the evaluation of any two (homogeneous) polynomials of the form $\sum_{i=0}^{n} c_i x^i y^{n-i}$, which have no common factor. It is conjectured that a carefully stated induction may be able to prove that at least $2 \ell og\ n$ multiplications are needed to perform such a computation.

Evaluation of a Homogeneous Polynomial of Degree n

The active operation arguments, noted earlier in this paper, can be used to show that essentially one multiplication or division is required for each independent non constant term of a polynomial. That is, roughly, for each parameter, one such operation is required. If preconditioning is permitted, one multiplication is needed for every two parameters. We shall now demonstrate an extention of this concept and show that for a particular type of polynomial more multiplications and divisions are needed than there are "degrees of freedom", by simply adding the number of operations needed by active operation counting and those for simple growth arguments (at least $\ell og\ n$ multiplications are needed to find x^n).

Consider the polynomial in two variables

$$P(x,y) = \sum_{i=0}^{n} a(i)x^i y^n$$

which shall be referred to as a homogeneous bivariate polynomial of degree n. Using the exponentiation technique of the last section an essentially optimal method of evaluating this function may be obtained as we show the following result.

Theorem 4 - A general homogeneous polynomial of degree n in two variables can be evaluated in $n+\ell og\ n + O(\ell og\ n/\ell og\ \ell og\ n)$ multiplications and divisions and n additions. Furthermore, at least n additive operations and $n+\lfloor \ell og\ n \rfloor$ mult/div are needed. Hence, this method is almost optimal.

Proof - P(x,y) may be written as

$$P(x,y) = \sum_{i=0}^{n} a(i)x^i y^{n-i} = y^n \sum_{i=0}^{n} a(i)(x/y)^i \ .$$

Hence it may be evaluated in 1 division, $n+\ell og\ n + O(\ell og\ n/\ell og\ \ell og\ n)$ multiplications and n additions by evaluating y^n and $\sum_{i=0}^{n} a(i)(x/y)^i$. The essential optimality of this method may be shown by first proving a more general result.

Lemma - The evaluation of the functional form

$$P(x,y) = y^k(\sum_{i=0}^{n} L_i(\vec{a})(x/y)^i + r(x,y))$$

> in which there are $u > 0$ linearly independent $L_i(\vec{a})$'s and $r(x,y)$ is any rational function, requires at least $u+\lfloor \ell og\ k \rfloor$ mult/div not counting multiplication or division by constants.

Consider the case in which $r = 0$, and make substitution $a=x=y$. Then the function becomes y^{k+1}, and so requires at least $\lceil \ell og(k+1) \rceil$ mult/div.

Suppose the lemma is true for all $v < u$, and the first active operation in the evaluation of such a form with u independent $L_i(\vec{a})$'s is

$$(L'(\vec{a})+r'(x,y)) \quad \{\times,/\} \quad ((ca(i)+L''(a(0),\ldots\hat{a}(i),\ldots a(n))+r''(x,y))$$

where $c \neq 0$ is a constant (we use the convention $\hat{a}(i)$ to denote the absence of $a(i)$ as a parameter). Then, if we set

$$a(i) = 1-(L''(a(0),\ldots\hat{a}(i),\ldots a(n))+r''(x,y))/c$$

and substitute back into the definition of $P(x,y)$, we have an expression of the form

$$P'(x,y) = y^k(\sum_{i=0}^{n} L_i'(\vec{a})(x/y)^i+r*(x,y)) \quad ,$$

in which at least u-1 of the L'_i's are linearly independent. Therefore, by the induction hypothesis, at least $\lfloor \ell og\ k \rfloor + u-1$ mult/div are needed to evaluate $P'(x,y)$. Hence the evaluation of $P(x,y)$ requires at least 1 more, or $\lfloor \ell og\ k \rfloor + u$ multiplications and divisions of the type we are counting.

Q.E.D.

The theorem follows directly from the lemma. The optimality with respect to additive operations follows by letting $y = 1$.

Q.E.D.

Conclusion

We have studied the asymptotic behaviour of the number of arithmetics in computing certain polynomial forms. Several algorithms have been presented. Some of these, such as the second polynomial and derivative algorithm may be of practical value. Others, such as the full precision polynomial algorithm, for now, only suggest asymptotic behaviour and general techniques for developing efficient algorithms.

From the forms studied, and other considerations, it is quite apparent that the "optimality" or non-optimality of an algorithm depends very heavily on the details of the model of computation and the step counting function. In spite of the well-known optimality of Horner's rule in the usual setting of polynomial evaluation, as one departs from this setting many questions remain open. The reason is the same as in much of Computational Complexity. As soon as the difficulty appears to exceed the number of inputs, most known techniques are inadequate and for establishing lower bounds.

Acknowledgement

This work was supported by the National Research Council of Canada.

References

[1] Belaga, E.C., "Some Problems in the Computation of Polynomials", Dokl. Akad. Nauk. SSSR, 123 (1958), pp. 775-777.

[2] Borodin, A., "Horner's Rule is Uniquely Optimal", Theory of Machines and Computations, (Ed. Z. Kohari and A. Paz)(1971) pp. 45-48.

[3] Borodin, A., and I. Munro, "Evaluation of Polynomials at Many Points", Information Processing Letters, Vol. 1, No. 2, July 1971, pp. 66-68.

[4] Brauer, A., "On Addition Chains", Bull. AMS 45 (1939), pp. 736-739.

[5] Cooley, J.W., and J.W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Mathematics of Computation, Vol. 19, No. 90 (1965) pp. 297-301.

[6] Erdös, P., "Remarks on Number Theory III - On Addition Chains", Acta Arithmetica 6 (1960), pp. 77-81.

[7] Fiduccia, C.M., "Fast Matrix Multiplication", Doctoral dissertation, Brown University.

[8] Hopcroft, J., Personal Communication

[9] Kirkpatrick, D., "On the Additions Necessary to Compute Certain Functions", M.Sc. Thesis, Department of Computer Science, University of Toronto (1971).

[10] Motzkin, T.S., "Evaluation of Polynomials and Evaluation of Rational Functions", Bull. Amer. Math. Soc. 61 (1955), p. 165.

[11] Munro, I., "Some Results Concerning Efficient and Optimal Algorithms", Proc. Third Annual Symp. on Theory of Computing, May 1971, pp. 40-44.

[12] Munro, I., "Efficient Polynomial Evaluation," Proc. Sixth Annual Princeton Conference on Information Sciences and Systems, March 1972, to appear.

[13] Pan, V.Y., "Methods of Computing Values of Polynomials", Russian Mathematical Surveys, Vol. 21, No. 1, 1966.

[14] Paterson, M., and L. Stockmeyer, "Bounds on the Evaluation Time of Rational Functions", Proc. Twelfth Annual IEEE Symposium on Switching and Automata Theory, October 1971, pp. 140-143.

[15] Schönhage, A., and V. Strassen, "Fast Multiplication of Large Numbers", Computing Vol. 7 (1971) pp. 281-292.

[16] Strassen, V., "Gaussian Elimination is Not Optimal", Numerische Mathematik 13 (1969), pp. 354-356.

[17] Strassen, V., "Evaluation of Polynomials", Proc. IBM Symp. on Complexity of Computer Computations, May 1972, to appear.

[18] Winograd, S., "On the Number of Multiplications Necessary to Compute Certain Functions", Comm. Pure and Applied Math., Vol. 23, 1970, pp. 165-179.

[19] Winograd, S., "On the Multiplication of 2 by 2 Matrices", IBM Research Report RC 2767, January 1970.