FAST EVALUATION AND INTERPOLATION

H. T. Kung

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa.

January, 1973

This work was supported in part by the National Science Foundation under grant GJ-32111 and the Office of Naval Research under Contract N00014-67-A-0314-0010, NR 044-422.

ABSTRACT

A method for dividing a polynomial of degree (2n-1) by a precomputed nth degree polynomial in 0(n log n) arithmetic operations is given. This is used to prove that the evaluation of an nth degree polynomial at n+1 arbitrary points can be done in 0(n log^ n) arithmetic operations, and consequently, its dual problem, interpolation of an nth degree polynomial from n+1 arbitrary points can be performed in $0(n \log^2 n)$ arithmetic operations. The best previously known algorithms required 0(n log^ n) arithmetic operations.

1. INTRODUCTION

Given (x_i, y_i) $(0 \le i \le n)$, the interpolation problem is the determination of the coefficients $\{c_i\}$ $(0 \le i \le n)$ of the unique polynomial $P(x) = \sum_{0 \le i \le n} c_i x^i$ of degree $\le n$ such that $P(x_i) = y_i$ $(0 \le i \le n)$. If a classical method such as the Lagrange or Newton formula is used, interpolation takes $O(n^2)$ operations. (In this paper all arithmetic operations will be counted. We simply write operations to denote arithmetic operations.) However, Horowitz (1972) has shown that interpolation can be done in $O(n \log^3 n)$ operations by using the Fast Fourier Transform (FFT), and he has shown that interpolation is reducible to evaluation of an nth degree polynomial at n+1 points. Moenck and Borodin (1972) have shown that the evaluation problem is reducible to the division problem, and they have shown that both evaluation and interpolation can be done in $O(n \log^3 n)$ operations, and precomputed interpolation (knowing the x_i in advance) can be performed in $O(n \log^2 n)$ operations. The purpose of this paper is to show that, without using any precomputation, both evaluation and interpolation can be done in $O(n \log^2 n)$ operations. As a corollary we show that an nth degree polynomial and all its derivatives can be evaluated at any point in $O(n \log^2 n)$ operations.

We shall use the same approach as used by Moenck and Borodin (1972). But we shall first precompute all necessary divisors in $O(n \log^2 n)$ operations so that each division can be done in $O(n \log n)$ operations. This results in faster evaluation and faster interpolation.

After the work reported here was completed, the author received a report from V. Strassen, entitled, "Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten". Using different techniques Strassen proves that interpolation can be done in $O(n \log n)$ multiplications or divisions and he states that his techniques can be used to prove that interpolation can be done in $O(n \log^2 n)$ arithmetic operations.

2. PRELIMINARIES

We shall work over the field of complex numbers.

Theorem 2.1. (Fast Polynomial Multiplication)

Let $A(x) = \sum_{0\le i\le n} a_i x^i$ and $B(x) = \sum_{0\le i\le n-1} b_i x^i$ be any two polynomials. Let $A(x)\cdot B(x) = \sum_{0\le i\le 2n-1} c_i x^i$. Then $\{c_i\}$ $(0 \le i \le 2n-1)$ can be obtained in $O(n \log n)$ operations.

Theorem 2.2.

Let $\{a_i\}$ $(0 \le i \le n)$ and $\{b_i\}$ $(0 \le i \le n-1)$ be any two sequences of numbers. Then

$$\begin{bmatrix} a_n & a_{n-1} & \cdots & a_1 \\ & \ddots & \ddots & \vdots \\ & & \ddots & a_{n-1} \\ & & & a_n \end{bmatrix} \begin{bmatrix} b_0 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{bmatrix} \tag{2.1}$$

can be computed in $O(n \log n)$ operations.

Proof.

Let $A(x) = \sum_{0\le i\le n} a_i x^i$ and $B(x) = \sum_{0\le i\le n-1} b_i x^i$. Suppose that $A(x)\cdot B(x) = \sum_{0\le i\le 2n-1} c_i x^i$. It is clear that the computation of (2.1) is equivalent to the computation of $\{c_i\}$ $(n \le i \le 2n-1)$. Thus the proof follows from Theorem 2.1. QED

Theorem 2.3.

For any sequence $\{a_i\}$ $(0 \le i \le n)$ of numbers with $a_n \ne 0$, let $\sum_{0\le i\le n-1} \bar{a}_i x^i$ be the unique polynomial $q(x)$ such that

$$x^{2n-1} = q(x)\cdot\left(\sum_{0\le i\le n} a_i x^i\right) + r(x),\ \deg r < n. \tag{2.2}$$

Then, we have that

$$\begin{bmatrix} a_n & a_{n-1} & \cdots & a_1 \\ & \ddots & \ddots & \vdots \\ & & \ddots & a_{n-1} \\ & & & a_n \end{bmatrix}^{-1} = \begin{bmatrix} \bar{a}_{n-1} & \bar{a}_{n-2} & \cdots & \bar{a}_0 \\ & \ddots & \ddots & \vdots \\ & & \ddots & \bar{a}_{n-2} \\ & & & \bar{a}_{n-1} \end{bmatrix} \tag{2.3}$$

and the sequence $\{\bar{a}_i\}$ $(0 \le i \le n-1)$ can be obtained in $O(n \log^2 n)$ operations.

Proof.

Let $(\sum_{0\le i\le n} a_i x^i)\cdot(\sum_{0\le i\le n-1} \bar{a}_i x^i) = \sum_{0\le i\le 2n-1} c_i x^i$. Then $\sum_{0\le i\le 2n-1} c_i x^i = x^{2n-1} - r(x)$. Since $\deg r < n$, $c_{2n-1} = 1$ and $c_i = 0$ for $i = n, n+1, \ldots, 2n-2$. Therefore,

$$\begin{bmatrix} a_n & a_{n-1} & \cdots & a_1 \\ & \ddots & \ddots & \vdots \\ & & \ddots & a_{n-1} \\ & & & a_n \end{bmatrix} \begin{bmatrix} \bar{a}_0 \\ \vdots \\ \bar{a}_{n-2} \\ \bar{a}_{n-1} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{2.4}$$

Furthermore, from (2.4), one can easily show that, for any i $(1 \le i \le n)$,

$$\begin{bmatrix} a_n & a_n & \cdots & a_1 \\ & \ddots & \ddots & \vdots \\ & & \ddots & \vdots \\ & & \ddots & a_{n-1} \\ & & & a_n \end{bmatrix} \begin{bmatrix} \bar{a}_{n-i} \\ \vdots \\ \bar{a}_{n-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \longleftarrow \text{ith component}$$

This proves (2.3). By using the fast division algorithm given by Moenck and Borodin (1972), the unique polynomial q(x) (i.e., the sequence $\{\bar{a}_i\}$ $(0 \le i \le n-1)$) can be computed in $O(n \log^2 n)$ operations. QED

Definition 2.4. (Precomputing)

Given any polynomial $P(x) = \sum_{0 \le i \le n} a_i x^i$ with $a_n \neq 0$, by precomputing P(x), we shall mean the computation of the $\{\bar{a}_i\}$ $(0 \le i \le n-1)$ which are defined by (2.2) or (2.3). That is, precomputing P(x) is just the division of x^{2n-1} by P(x).

Hence, by Theorem 2.3, we can precompute an nth degree in $O(n \log^2 n)$ operations. Since this bound will be sufficient to prove the results in this paper, no attempt has been made to improve it.

3. FAST DIVISION USING PRECOMPILED DIVISOR

Theorem 3,1.

Let U(x) = S u.x and V(x) = £ v.x (v ^ 0). Suppose that
0£i£2n-1 0fi^n
V(x) has already been precomputed, i.e., {^} (0 ^ i ^ n-1) are available with no associated cost. Then we can compute the unique polynomials Q(x) and R(x) such that

(3.1) U(x) « Q(x)-V(x) + R(x), deg R < n

in 0(n log n) operations.

Proof.

It suffices to show that to compute Q(x) we only require 0(n log n) operations, since R(x) = U(x) - Q(x)«V(x) and Q(x)«V(x) can be computed in 0(n log n) operations by Theorem 2.1. Let Q(x) = q.x , and let
0<a<*1-1
Q(x)'V(x) = I. c.x'. From (3.1), it is clear that u. = c, for
0^i^2n-1
i = n,...,2n-1. Therefore,

v v - ••• v-
n n-1 1

n-1 V2

n *>n

and hence, by (2.3),

o n-2 n

n-2

1 ^n-
n-1

The theorem then follows from Theorem 2.2. QED

4. FAST EVALUATION

Moenck and Borodin (1972) have shown that evaluation is reducible to division and have proved the following theorem:

<u>Theorem 4.1</u>. (Moenck and Borodin (1972))

Let $U(x)$ be a polynomial of degree $n = 2^r-1$. Then we can evaluate $U(x)$ at $n+1$ <u>arbitrary</u> points $x_0,x_1,\dots,x_n$ in $O(g(n)\log n+f(n)\log n)$ operations, provided that we can divide a polynomial of degree $(2n-1)$ by an nth degree polynomial in $O(g(n))$ operations and multiply two nth degree polynomials in $O(f(n))$ operations.

This fast evaluation algorithm requires certain divisions. The divisors are exactly the members of the following family except the polynomial at level r+1.

$$
(4.1)\qquad
\begin{array}{cl}
x-x_0,\ x-x_1,\ x-x_2,\ x-x_3,\quad \dots\quad ,\ x-x_{n-1},\ x-x_n & \text{Level } 1\\
(x-x_0)(x-x_1),\ (x-x_2)(x-x_3),\quad \dots\quad ,\ (x-x_{n-1})(x-x_n) & \text{Level } 2\\
\prod_{i=0}^{3}(x-x_i),\quad \dots\quad \prod_{i=n-3}^{n}(x-x_i) & \text{Level } 3\\
\ddots\qquad\qquad ⋰ & \\
\prod_{i=0}^{2^{r-1}-1}(x-x_i)\qquad \prod_{i=2^{r-1}}^{2^r-1}(x-x_i) & \text{Level } r\\
\prod_{i=0}^{2^r-1}(x-x_i) & \text{Level } r+1
\end{array}
$$

<u>Theorem 4.2</u>.

All polynomials in (4.1) can be precomputed in $O(n\log^2 n)$ operations.

Proof.

We first convert all polynomials in (4.1) into the form $\sum_i h_i x^i$. This can be done in $O(n \log^2 n)$ operations (see Horowitz (1972)). Then we shall precompute the polynomials at level j from the precomputed polynomials at level j+1, for $j = r, r-1, \ldots, 1$. By Theorem 2.3, we can precompute the polynomial at level r+1 in $O(n \log^2 n)$ operations. Suppose that all polynomials at level j+1 have been precomputed. Let $D(x) = \sum_{0 \le i \le 2^j} d_i x^i$ be a polynomial at level j+1, and let $E(x) = \sum_{0 \le i \le 2^{j-1}} e_i x^i$ and $F(x) = \sum_{0 \le i \le 2^{j-1}} f_i x^i$ be those two polynomials such that $D(x) = E(x) \cdot F(x)$. By (2.2), we know that

$$x^{2^{j+1}-1} = (\sum_{0 \le i \le 2^j - 1} \bar{d}_i x^i) \cdot D(x) + r_D(x), \quad \deg r_D < 2^j.$$

Since $D(x) = E(x) \cdot F(x)$, it follows that

$$\frac{x^{2^j-1}}{E(x)} = \frac{(\sum_{0 \le i \le 2^j - 1} \bar{d}_i x^i) \cdot F(x)}{x^{2^j}} + \frac{r_D(x)}{x^{2^j} E(x)} \quad .$$

But, by (2.2),

$$\frac{x^{2^j-1}}{E(x)} = \sum_{0 \le i \le 2^{j-1}-1} \bar{e}_i x^i + \frac{r_E(x)}{E(x)} \quad , \deg r_E < 2^{j-1} \; .$$

Hence, if $(\sum_{0 \le i \le 2^j - 1} \bar{d}_i x^i) \cdot F(x) = \sum_{0 \le i \le 2^j + 2^{j-1} - 1} g_i x^i$, then

$$\sum_{0 \le i \le 2^{j-1}-1} g_{i+2^j} x^i + \frac{\sum_{0 \le i \le 2^j - 1} g_i x^i}{x^{2^j}} + \frac{r_D(x)}{x^{2^j} E(x)} = \sum_{0 \le i \le 2^{j-1}-1} \bar{e}_i x^i + \frac{r_E(x)}{E(x)} .$$

By the uniqueness of the partial fraction expansion, it is easy to see that

$$\bar{e}_i = g_{i+2^j} \text{ for all } i = 0,1,\ldots,2^{j-1}-1.$$

Therefore, we can precompute E(x) by computing $(\sum_{0\le i\le 2^j-1} \bar{d}_i x^i)\cdot F(x)$, which can be performed in $O(j\cdot 2^j)$ operations by Theorem 2.1. Similarly, we can precompute F(x) in $O(j\cdot 2^j)$ operations. Since there are $\frac{2^r}{2^{j-1}}$ polynomials at level j, all polynomials at level j can be precomputed in $O(\frac{2^r}{2^{j-1}}\cdot j\cdot 2^j) = O(j\cdot 2^{r+1})$ operations. Hence, all polynomials in (4.1) can be precomputed in $O(\sum_{1\le j\le r} j\cdot 2^{r+1}) = O(r^2\cdot 2^r) = O(n \log^2 n)$ operations. QED

<u>Theorem 4.3.</u>

Let U(x) be a polynomial of degree $n = 2^r-1$. Then we can evaluate U(x) at n+1 <u>arbitrary</u> points $x_0,x_1,\ldots,x_n$ in $O(n \log^2 n)$ operations.

<u>Proof.</u>

We first precompute all divisors needed for the algorithm of Theorem 4.1. By Theorem 4.2, this takes $O(n \log^2 n)$ operations. Then by Theorem 3.1, all divisions used in the algorithm of Theorem 4.1 can be performed in $O(n \log n)$ operations. The proof follows from Theorem 4.1 by letting $g(n) = f(n) = n \log n$. QED

5. FAST INTERPOLATION

Horowitz (1972) has shown that interpolation is reducible to fast evaluation.

Theorem 5.1. (Horowitz (1972))

x*

Given n+1 = 2 pairs of numbers (x^y^) (0 ^ i ^ n), the coefficients of the unique polynomial P(x) of degree ^ n such that y^ = P(x^) (0 £ i £ n) can be obtained in 0(h(n) + f(n)log n) operations, provided that evaluation at n+1 point is 0(h(n)) operations and multiplication is 0(f(n)) operations.

Theorem 5.2.

Given n+1 = 2 pairs of points (x^,y^) (0 £ i £ n), the coefficients of the unique polynomial P(x) of degree ^ n such that y = P(x^) (0 ^ i ^ n)

2

can be obtained in 0(n log n) operations.

Proof.

Apply the result of Theorem 4.3 to Theorem 5.1. QED

Corollary 5.3.

An nth degree polynomial and all its derivatives can be evaluated at

2

any point in 0(n log n) operations.

Proof.

Suppose that we want to evaluate the nth degree polynomial P(x) and all its derivatives at some point a. Then it suffices to show that {d^} (0 ^ i ^ n)

2

such that P(x) = S d.(x-cy) , can be obtained in 0(n log n) operations.

0^i^n

First, we evaluate $P(x)$ at $n+1$ arbitrary distinct points $x_0, x_1, \ldots, x_n$. This takes $O(n \log^2 n)$ operations by Theorem 4.3. Next, we determine $\{d_i\}$ $(0 \le i \le n)$ such that $\sum_{i=0}^{n} d_i y_j^i = P(x_j)$, $y_j = x_j - \alpha$ for $j = 0,1,\ldots,n$. This is an interpolation problem and takes $O(n \log^2 n)$ operations by Theorem 5.2. QED

ACKNOWLEDGMENT

The author wishes to express his appreciation to his advisor, Professor J. F. Traub, for providing information on the subject and for his comments on this paper.

6. BIBLIOGRAPHY

E. Horowitz (1972), "A Fast Method for Interpolation Using Preconditioning," Information Processing Letters 1, pp. 157-163.

R. Moenck and A. Borodin (1972), "Fast Modular Transform Via Division," Proceedings of 13th Symposium on Switching and Automata Theory, pp. 90-96.