SIAM J. COMPUT.
Vol. 4, No. 4, December 1975

EVALUATING POLYNOMIALS AT FIXED SETS OF POINTS*

A. V. AHO†, K. STEIGLITZ‡ AND J. D. ULLMAN‡

Abstract. We investigate the evaluation of an $(n-1)$st degree polynomial at a sequence of n points. It is shown that such an evaluation reduces directly to a simple convolution if and only if the sequence of points is of the form $b, ba, ba^2, \cdots, ba^{n-1}$ for complex numbers a and b (the so-called "chirp transform"). By more complex reductions we develop an $O(n \log n)$ evaluation algorithm for sequences of points of the form

$$b+c+d, \quad ba^2+ca+d, \quad ba^4+ca^2+d, \cdots$$

for complex numbers a, b, c and d. Finally we show that the evaluation of an $(n-1)$st-degree polynomial and all its derivatives at a single point requires at most $O(n \log n)$ steps.

Key words. polynomial evaluation, derivative, fast Fourier transform, chirp transform, straight-line code, computational complexity, Taylor series

1. Introduction.

We consider the following problem. Given an infinite sequence of points $a_0, a_1, a_2, \cdots$, how long, as a function of n, does it take to evaluate an arbitrary dense univariate polynomial of degree $n-1$ at the first n of these points?

Our model of computation is the *straight-line code* model. For each n, we assume that the computation is performed by a sequence of assignment statements of the form $A \leftarrow B\ \theta\ C$, where A, B and C are variable names, constants, or the names of coefficients of the polynomial (input variables), and θ is one of the operators $+$, $-$, $\times$ or $/$. In addition, n variables are designated as output variables, and after execution of the sequence of assignment statements, these variables are to hold the values of the polynomial at the n points. Such a sequence of assignment statements will be called an *algorithm*, and the *complexity* of an algorithm is the number of assignment statements therein.

A straight-line algorithm that evaluates any $(n-1)$st-degree polynomial at n points is said to be an *evaluation algorithm*. The inputs to the algorithm are the coefficients of the polynomial. A sequence of points $a_0, a_1, a_2, \cdots$ is said to be of *complexity* at most $T(n)$, if for all positive n there is a straight-line algorithm with at most $T(n)$ statements that evaluates any $(n-1)$st-degree polynomial at the first n points of the sequence.

It is known that an arbitrary sequence of points is of complexity at most $O(n \log^2 n)$([1] modified by the treatment in [2], [3], [4]). Certain sequences of points, however, are of complexity at most $O(n \log n)$. The best known such sequence of points is the "chirp transform" [5], [6], a generalization of the "fast

* Received by the editors May 13, 1974, and in revised form November 15, 1974. The work of the second author was supported in part by the U.S. Army Research Office, Durham, under Contract DAHCO4-09-C-0012, and by The National Science Foundation under Grant GJ-965. The work of the third author was supported in part by The National Science Foundation under Grant GJ-35570.

† Bell Laboratories, Murray Hill, New Jersey 07974.

‡ School of Engineering and Applied Science, Princeton University, Princeton, New Jersey 08540.

Fourier transform" (FFT) [7]. In the chirp transform, the sequence of points z^0, z^1, z^2, $\cdots$ is used, where z is any complex number. A polynomial $\sum_{i=0}^{n-1} b_i x^i$ can be evaluated at the points z^0, z^1, $\cdots$, z^{n-1} in $O(n \log n)$ time as follows. To compute

$$c_j = \sum_{i=0}^{n-1} b_i z^{ij} \quad \text{for } 0 \leqq j \leqq n-1, \tag{1}$$

we rewrite (1) as:

$$c_j = \sum_{i=0}^{n-1} b_i \, z^{-(j-i)^2/2} \, z^{i^2/2} \, z^{j^2/2} \quad \text{for } 0 \leqq j \leqq n-1. \tag{2}$$

Equation (2) is a convolution which can be evaluated in $O(n \log n)$ steps with the FFT.

Our goal is to increase the set of sequences which are known to have $O(n \log n)$ evaluation algorithms. While an interesting problem in its own right, digital signal processing provides the practical motivation for considering the evaluation of high-degree dense polynomials at more general sequences of points than that of the chirp transform. See [8] and [9], for example.

The approach we use is to consider *classes* of sequences. The $T(n)$ class of sequences is the set of all sequences which have an $O(T(n))$ evaluation algorithm. Thus for each complex number z, the sequence z^0, z^1, $\cdots$ is in the $n \log n$ class, and every sequence is in the $n \log^2 n$ class. We shall ultimately consider closure properties of classes, but first we shall consider to what extent the chirp transformation generalizes directly.

It should be noted that our definition of "class" smears the boundary between sequences of distinct degrees of difficulty. For example, it is by no means clear that there is speedup by constant factors in the straight-line code model, as there is for Turing machines [10]. It is likely that there are sequences that can be evaluated in time $(1+\varepsilon)T(n)$ but not in time $T(n)$ for any reasonable $T(n)$ and $\varepsilon > 0$. Nevertheless, the definition is a useful one to make, and we shall use it to advantage subsequently.

2. Uniqueness of the chirp transform. We have seen that the chirp transform reduces to a convolution of the form $\alpha(j) \sum_{i=0}^{n-1} b_i \beta(i) \gamma(j-i)$, where the b_i's are the coefficients of the polynomial to be evaluated, and α, β and γ are independent of the b_i's. It is interesting to consider what other transformations, if any, can be reduced to a convolution of this form. The following theorem shows that except for a constant factor, the chirp transform is the most we can obtain by this technique.

THEOREM 1. *For $n \geqq 3$, suppose that the evaluation of an arbitrary $(n-1)$-st-degree polynomial $\sum_{i=0}^{n-1} b_i x^i$ at the points a_0, a_1, $\cdots$, a_{n-1} can be expressed as a convolution of the form*

$$\sum_{i=0}^{n-1} b_i a_j^i = \alpha(j) \sum_{i=0}^{n-1} b_i \beta(i) \gamma(j-i) \quad \text{for } 0 \leqq j \leqq n-1 \tag{3}$$

for some functions α, β and γ independent of the b's. Then $a_j = z_1(z_2)^j$ for some complex numbers z_1 and z_2.

Proof. Since the b_i's are arbitrary, the left and right sides of (3) must agree term by term. Thus

$$a_j^i = \alpha(j)\beta(i)\gamma(j-i) \tag{4}$$

for all i and j between 0 and $n-1$.

Suppose temporarily that $a_j \neq 0$ for any j. Taking logarithms of (4), we obtain

$$i \log a_j = \log \alpha(j) + \log \beta(i) + \log \gamma(j-i) \quad \text{for } 0 \leqq i, j \leqq n-1. \tag{5}$$

Taking finite differences of (5) with respect to $i+1$ and j gives

$$\log a_j = \log \frac{\beta(i+1)}{\beta(i)} + \log \frac{\gamma(j-i-1)}{\gamma(j-i)} \tag{6}$$

for $0 \leqq i < n-1$ and $0 \leqq j \leqq n-1$, and

$$i \log \frac{a_j}{a_{j-1}} = \log \frac{\alpha(j)}{\alpha(j-1)} + \log \frac{\gamma(j-i)}{\gamma(j-i-1)} \tag{7}$$

for $0 \leqq i \leqq n-1$ and $0 < j \leqq n-1$. Summing (6) and (7), we obtain

$$\log a_j + i \log \frac{a_j}{a_{j-1}} = \log \frac{\alpha(j)}{\alpha(j-1)} + \log \frac{\beta(i+1)}{\beta(i)} \tag{8}$$

for $0 \leqq i < n-1$ and $0 < j \leqq n-1$. Taking (8) at $i = 1$ and subtracting from it (8) at $i = 0$, we obtain

$$\log \frac{a_j}{a_{j-1}} = \log \frac{\beta(0)\beta(2)}{\beta(1)^2} \quad \text{for } 0 < j \leqq n-1. \tag{9}$$

(Note that $n \geqq 3$ is necessary for this step to make sense.) It follows from (9) that

$$\frac{a_j}{a_{j-1}} = \frac{\beta(0)\beta(2)}{\beta(1)^2} \quad \text{for } 0 < j \leqq n-1,$$

and therefore

$$a_j = a_0 \left[\frac{\beta(0)\beta(2)}{\beta(1)^2} \right]^j \quad \text{for } 0 \leqq j \leqq n-1.$$

Let $z_1 = a_0$ and $z_2 = \beta(0)\beta(2)/\beta(1)^2$ to prove the theorem.

The detail which remains is what happens when $a_j = 0$ for some j, say j_0. Referring back to (3), we see that the left side evaluated at a_{j_0} is just b_0. Thus, in place of equation (4) with $j = j_0$, we have

$$\alpha(j_0)\beta(i)\gamma(j_0-i) = 0 \quad \text{for } 1 \leqq i \leqq n-1, \tag{10}$$

$$\alpha(j_0)\beta(0)\gamma(j_0) = 1. \tag{11}$$

From (11) we see $\alpha(j_0) \neq 0$. Thus by (10) we have $\beta(i)\gamma(j_0-i) = 0$ for $1 \leqq i \leqq n-1$.

The theorem is easily seen to hold if $a_j = 0$ for all j. Thus assume the contrary. If $\beta(i) = 0$ for any i, then the right side of (3) is independent of b_i. This is impossible, since we assumed not all a_j's are zero. We may conclude that $\gamma(j_0-i) = 0$ for $1 \leqq i \leqq n-1$. It is sufficient to consider two cases.

Case 1. $a_{j_0}=0$ and $a_{j_0+1}\neq 0$. Then, since $\gamma(j_0-1)=0$, it follows that $\gamma((j_0+1)-2)=0$. Thus, for $j=j_0+1$, the right side of (3) is independent of b_2. Since $n\geqq 3$ is assumed, we have a contradiction.

Case 2. $a_{j_0}=0$ and $a_{j_0-1}\neq 0$. Then the right side of (3) is independent of b_0 for $j=j_0-1$, again yielding a contradiction. □

Note that Theorem 1 is trivially true for $n=1$, but there is a counterexample with $a_0=0$ for the case $n=2$.

3. Closure properties of sequence classes. We see from Theorem 1 that the chirp transform is essentially all that we can obtain by a simple convolution. More extensive algebraic manipulations, however, do yield larger classes of sequences for the n log n class. Before looking at these more complex operations, we derive several "closure properties" that hold for the various sequence classes.

LEMMA 1. *If sequence* $a_0, a_1, a_2, \cdots$ *is in class* $T(n)\geqq n$, *and* c *is any complex number, then sequence* $ca_0, ca_1, ca_2, \cdots$ *is also in class* $T(n)$.

Proof. Let $\mathbf{A}_n$ be an algorithm that takes as input the coefficients $b_0, b_1, \cdots, b_{n-1}$ and produces $d_j=\sum_{i=0}^{n-1} b_i a_j^i$ for $0\leqq j\leqq n-1$ as outputs. Then we may construct algorithm $\mathbf{B}_n$ to compute $e_j=\sum_{i=0}^{n-1} b_i(ca_j)^i$ for $0\leqq j\leqq n-1$. $\mathbf{B}_n$ works as follows:

1. In $n-2$ steps, compute $f_i=c^i$ for $2\leqq i\leqq n-1$. Let $f_0=1$ and $f_1=c$.
2. In $n-1$ steps, compute $g_i=b_i f_i$ for $0\leqq i\leqq n-1$.
3. Apply algorithm $\mathbf{A}_n$ to coefficients $g_0, g_1, \cdots, g_{n-1}$.
4. The outputs of $\mathbf{A}_n$ are the desired outputs for $\mathbf{B}_n$.

It should be clear that $\mathbf{B}_n$ works, and that the length of the straight-line algorithm $\mathbf{B}_n$ is $2n-3$ plus the length of $\mathbf{A}_n$. Thus, since $T(n)\geqq n$, we know that the length of $\mathbf{B}_n$ does not exceed $3T(n)$. □

LEMMA 2. *If* $a_0, a_1, a_2, \cdots$ *is in the* $T(n)\geqq n$ log n *class, and* k *is any positive integer, then* $a_0^k, a_1^k, a_2^k, \cdots$ *is also in the* $T(n)$ *class.*

Proof. The proof is again straightforward. Given the coefficients $b_0, b_1, \cdots, b_{n-1}$, we construct a new sequence of coefficients of length kn by inserting $k-1$ 0's after each of the b's. Then we break this sequence into k subsequences of length n. We let the subsequences represent k polynomials $p_0, p_1, \cdots, p_{k-1}$. We now have

$$\sum_{i=0}^{n-1} b_i a_j^{ki}=\sum_{r=0}^{k-1} a_j^{rn} p_r(a_j) \tag{12}$$

for $0\leqq j\leqq n-1$.

We use the assumed $O(T(n))$ algorithm k times to evaluate the p_r's at $a_0, a_1, \cdots, a_{n-1}$. The terms a_j^m for $0\leqq j\leqq n-1$ and $0\leqq r\leqq k-1$ can be evaluated in $O(kn+n \log n)$ steps, and the right side of (12) can be evaluated in $O(kn)$ steps given the a_j^{rn}'s and $p_r(a_j)$'s. Thus the entire algorithm requires $O(kT(n))+O(kn+n \log n)$ steps. Since k is a constant and $T(n)\geqq n \log n$, this function is of the order of $T(n)$. □

LEMMA 3. *If* $a_0, a_1, \cdots$ *is in the* $T(n)\geqq n$ log n *class and* c *is any complex number, then* $a_0+c, a_1+c, \cdots$ *is also in the* $T(n)$ *class.*

Proof. We wish to compute $\sum_{i=0}^{n-1} b_i(a_j+c)^i$ for $0 \leqq j \leqq n-1$. This can be done in the following manner:

$$\begin{aligned}\sum_{i=0}^{n-1} b_i(a_j+c)^i &= \sum_{i=0}^{n-1} b_i \sum_{r=0}^{i} \binom{i}{r} a_j^r c^{i-r} = \sum_{r=0}^{n-1}\sum_{i=r}^{n-1} b_i \binom{i}{r} a_j^r c^{i-r} \\ &= \sum_{r=0}^{n-1} \frac{a_j^r}{r!} \sum_{i=r}^{n-1} \frac{b_i i! c^{i-r}}{(i-r)!}.\end{aligned} \tag{13}$$

If we define $f(x) = b_x x!$ for $0 \leqq x \leqq n-1$ and

$$g(x) = \begin{cases} c^{-x}/(-x)! & \text{for } -(n-1) \leqq x \leqq 0, \\ 0 & \text{for } 1 \leqq x \leqq n-1, \end{cases}$$

then we can allow the inner summation of (13) to range from 0 to $n-1$ and write (13) as:

$$\sum_{i=0}^{n-1} b_i(a_j+c)^i = \sum_{r=0}^{n-1} \frac{a_j^r}{r!} \sum_{i=0}^{n-1} f(i)g(r-i). \tag{14}$$

It is easy to see how to compute the necessary values of $f(x)$ and $g(x)$ in $O(n)$ steps. Then the inner summation of (14) can be evaluated for $0 \leqq r \leqq n-1$ in $O(n \log n)$ steps, since it is a convolution. In $O(n)$ more steps, we can compute $r!$ for $0 \leqq r \leqq n-1$. Thus we can compute $d_r = (1/r!)\sum_{i=0}^{n-1} f(i)g(r-i)$ for $0 \leqq r \leqq n-1$ in $O(n \log n)$ steps.

Thus the problem of evaluating $\sum_{i=0}^{n-1} b_i x^i$ at points a_0+c, $a_1+c, \cdots$ has been reduced in $O(n \log n)$ steps to the problem of evaluating $\sum_{i=0}^{n-1} d_i x^i$ at points $a_0, a_1, \cdots$. The latter evaluation can be done in $T(n)$ steps. Since $T(n) \geqq n \log n$, the desired evaluation takes $O(T(n))$ steps. □

We may now combine the three lemmas to obtain additional closure properties of sequence classes.

THEOREM 2. *If $a_0, a_1, \cdots$ is in class $T(n) \geqq n \log n$, b and c are complex numbers, and k is any positive integer, then $ba_0^k+c, ba_1^k+c, \cdots$ is in class $T(n)$.*

Proof. By Lemma 2, sequence $a_0^k, a_1^k, \cdots$ is in class $T(n)$. By Lemma 1, so is sequence $ba_0^k, ba_1^k, \cdots$, and by Lemma 3 we have the theorem. □

THEOREM 3. *If sequence $a_0, a_1, \cdots$ is in class $T(n) \geqq n \log n$, and b, c and d are complex numbers, then $ba_0^2+ca_0+d, ba_1^2+ca_1+d, \cdots$ is in class $T(n)$.*

Proof. By completing the square, we can find complex numbers e and f such that for all x,

$$b(x+e)^2+f = bx^2+cx+d.$$

By Lemma 3, $a_0+e, a_1+e, \cdots$ is in class $T(n)$. Using Theorem 2 with $k=2$, we see that $b(a_0+e)^2+f, b(a_1+e)^2+f, \cdots$ is in class $T(n)$. This sequence is the desired one. □

We have the following corollary to the theorems above and the chirp transform theorem.

THEOREM 4. *The following sequences are in class $n \log n$ for complex numbers a, b, c and d, and positive integer k:*

$$b+c, ba^k+c, ba^{2k}+c, \cdots \tag{15}$$

and

$$b+c+d,\ ba^2+ca+d,\ ba^4+ca^2+d,\ \cdots. \tag{16}$$

Note that (15) is a special case of (16) with a, b and d set to a^k, 0 and b, respectively.

4. Evaluation of a polynomial and all its derivatives. There has been recent interest in the question of how fast one can evaluate a polynomial and all its derivatives at a single point. Shaw and Traub [11] show that one can reduce the number of multiplications to $O(n)$, although the algorithm given required $O(n^2)$ total operations. Kung [3] and Borodin and Munro [13] independently observed that evaluation of a polynomial and its derivatives reduces to evaluation and interpolation of polynomials, and thus could be done in $O(n \log^2 n)$ steps. Kung [12] gives another $O(n \log^2 n)$ algorithm without using evaluation and interpolation. In this section we show that problem can be done in $O(n \log n)$ steps. This result hinges upon the following definition and lemma.

The *Taylor series* of a polynomial $\sum_{i=0}^{n-1} b_i x^i$ at point a is that polynomial $\sum_{i=0}^{n-1} c_i x^i$ such that for all x,

$$\sum_{i=0}^{n-1} c_i (x-a)^i = \sum_{i=0}^{n-1} b_i x^i.$$

LEMMA 4. *The problems of evaluating a polynomial and all its derivatives at a point and of finding the Taylor series of a polynomial at a point require within* $2n$ *operations of each other for polynomials of degree* $n-1$.

Proof. Let $\sum_{i=0}^{n-1} b_i x^i = \sum_{i=0}^{n-1} c_i (x-a)^i$ for all x. Then

$$\frac{d^k}{dx^k}\left(\sum_{i=0}^{n-1} b^i x^i\right)\Bigg|_{x=a} = k!c_k.$$

Thus the kth derivative at point a and the kth coefficient of the Taylor expansion can be recovered from one another by multiplying or dividing by $k!$ □

THEOREM 5. *An* $(n-1)$*-st-degree polynomial and all its derivatives can be evaluated at point* c *in* $O(n \log n)$ *steps.*

Proof. In Lemma 3 we showed how to compute from c and the b_i's in $O(n \log n)$ steps those numbers d_j, $0 \leqq j \leqq n-1$, such that for all x,

$$\sum_{i=0}^{n-1} b_i (x+c)^i = \sum_{r=0}^{n-1} d_r x^r. \tag{17}$$

If (17) holds, then it is surely true that for any x,

$$\sum_{i=0}^{n-1} b_i x^i = \sum_{i=0}^{n-1} d_i (x-c)^i.$$

The theorem then follows from Lemma 4. □

Theorem 5 has been independently shown by Vari [14].

Acknowledgment. The authors would like to thank the referees for a number of perceptive comments.

REFERENCES

[1] R. MOENCK AND A. B. BORODIN, *Fast modular transforms via division*, Conf. Rec. IEEE 13th Ann. Symp. on Switching and Automata Theory, 1972, pp. 90–96.

[2] A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.

[3] H. T. KUNG, *Fast evaluation and interpolation*, Tech. Rep., Dept. of Computer Sci., Carnegie-Mellon Univ., Pittsburgh, 1973.

[4] M. SIEVEKING, *An algorithm for the division of power series*, Computing, 10 (1972), pp. 153–156.

[5] L. I. BLUESTEIN, *A linear filtering approach to the computation of the discrete Fourier transform*, IEEE Trans. Electroacoustics, AU-18 (1970), pp. 451–455.

[6] L. R. RABINER, R. W. SCHAFER AND C. M. RADER, *The chirp z-transform and its applications*, Bell System Tech. J., 48 (1969), pp. 1249–1292.

[7] J. M. COOLEY AND J. W. TUKEY, *An algorithm for the calculation of Fourier series*, Math. Comp., 19 (1965), pp. 297–301.

[8] A. OPPENHEIM AND D. H. JOHNSON, *Discrete representation of signals*, Proc. IEEE, 60 (1972), pp. 681–691.

[9] A. OPPENHEIM, D. H. JOHNSON AND K. STEIGLITZ, *Computation of spectra with unequal resolution using the fast Fourier transform*, Proc. IEEE (Lett), 59 (1971), pp. 299–301.

[10] J. HARTMANIS AND R. E. STEARNS, *On the computational complexity of algorithms*, Trans. Amer. Math. Soc., 117 (1965), 285–306.

[11] M. SHAW AND J. F. TRAUB, *On the number of multiplications for the evaluation of a polynomial and some of its derivatives*, J. Assoc. Comput. Mach., 21 (1974), pp. 161–167.

[12] H. T. KUNG, *A new upper bound on the complexity of derivative evaluation*, Information Processing Letters, 2 (1973), pp. 146–147.

[13] A. B. BORODIN AND I. MUNRO, *Notes on Efficient and Optimal Algorithms*, American-Elsevier, New York, 1975.

[14] T. M. VARI, *Some complexity results for a class of Toeplitz matrices*, Tech. Rep., Dept. of Computer Sci. and Mathematics, York Univ., Toronto, 1974.