REFERENCES

B82 Bini, D., "Parallel Solution of Certain Toeplitz Linear Systems", Tech. Report, University of Pisa, April 1982, and *SIAM J. on Computing*, Vol. 13, 2, pp. 268-276, 1984.

BP86 Bini, D. and V. Pan, "Polynomial Division and Its Computational Complexity", *J. of Complexity*, vol. 2, pp. 179-203, 1986.

BPa Bini, D., and V. Pan, "Numerical and Algebraic Computations with Matrices and Polynomials", manuscript, 1990.

BPb Bini, D., and V. Pan, "Parallel Polynomial Computations by Recursive Processes", *Proc. 2nd ACM Intern. Symp. on Symbolic and Algebraic Computations* (ISSAC-90), 1990.

BPc Bini, D., and V. Pan, "Improved Parallel Polynomial Division", manuscript, 1990.

EG Eppstein, D. and Z. Galil, "Parallel Algorithmic Techniques for Combinatorial Computation", *Annual Review of Computer Science*, vol. 3, pp. 233-283, 1988.

KR Karp, R. and V. Ramachandran, "A Survey of Parallel Algorithms for Shared Memory Machines", Report No. UCB/CSD 88/408, Computer Science Division, University of California, Berkeley, CA, 1988.

Sc82 Schönhage, S., "Asymptotically Fast Algorithms for the Numerical Multiplication and Division of Polynomials with Complex Coefficients", *Proc. EUROCAM*, Marseille, 1982.

Toom 63 Toom, A.L., "The Complexity of a Scheme of Functional Elements Realizing the Multiplication of Integers", *Soviet Math. Doklady*, vol. 3, pp. 714-716, 1963.

and it suffices to choose $H > 2n \max_\ell |t_\ell|$ in order to ensure (3.1).

We may let $a = 4$ (say), choose H according to the above rules, and then easily estimate the precision of computation that would ensure that $|q_i^* - q_i| < 0.5$ for all i. The resulting estimates for the precision of $O(m \log H)$ bits and for the parallel and sequential Boolean complexity of this computation are very close to the ones of [BP] for the algorithms of [B82] and [S82].

This should be no surprise because a more close examination immediately shows that our algorithm computes the same intermediate and output values as do the algorithms of [S82] and [B82] (compare [BP]), although the derivation and interpretation of the three algorithms are distinct.

$$T = \max \{|t_n|, |t_{n-1}|, \ldots, |t_{2n-m}|\} \ ,$$

which enables us to estimate

$$\max_{0 \le \ell < n} |r_\ell| \le \max_{0 \le \ell \le n} |t_\ell| \sum_{i=0}^{m-n} |q_i| + \max_{0 \le g \le m} |s_g|$$

in terms of the coefficients of $s(x)$ and $t(x)$ only and thus to obtain that

$$|q_i^* - q_i| \le 2nH^{-i} \left(\max_{0 \le \ell \le n} |t_\ell| (1+T)^{m-n} \sum_{g=n}^{m} |s_g| + \max_{0 \le g \le m} |s_g| \right) / (H-1) \ , \ \ i = 0, 1, \ldots, m-n \ .$$

Finally, we observe that the larger i, the stronger the bound (3.2) (provided that $H > 1$). Thus we may decrease the output error [without affecting the asymptotic complexity bound (2.1) by more than a constant factor] if we apply our algorithm to the polynomials $x^b s(x)$ [replacing $s(x)$] and $t(x)$ for a larger $b = O(m-n)$, say, for $b = m - n$. Then the original quotient $q(x)$ will turn into $\tilde{q}(x) = x^b q(x)$, and we will only need to use the output approximations to the $m - n + 1$ leading coefficients of this new quotient, whose approximation errors are bounded according to (3.2) for $i \ge b$.

In particular, let the coefficients of $s(x)$, $t(x)$, and consequently, $q(x)$ be integers; let $b = m - n$, H satisfy (3.1), and

$$H^b > 2an \max_\ell |r_\ell| \ ; \tag{3.3}$$

then

$$|q_i^* - q_i| < \frac{1}{a(H-1)} \quad \text{for all } i \ , \tag{3.4}$$

due to the bound (3.2), so that $H \ge 2$, and $|q_i^* - q_i| < 1/2$ for $a > 2$, and then the exact values of the q_i can be recovered from q_i^* by rounding-off to the nearest integers.

Due to Remark 3.1, we may satisfy (3.3) by choosing $b = m - n$ and

$$H = \left[2an \left((1+T)^{m-n} \sum_{g=n}^{m} |s_g| \max_{0 \le \ell \le n} |t_\ell| + \max_{0 \le g \le m} |s_g| \right) \right]^{1/(m-n)}$$

$$= O\left(\max_\ell |t_\ell| (an \max_g |s_g|)^{1/(m-n)} \right) \ ,$$

3. ERROR ESTIMATES AND AN IMPROVED ALGORITHM

Let us first estimate the approximation errors of our algorithm and then use these estimates to improve the algorithm.

Proposition 3.1. *Let*

$$H^n > 2\sum_{\ell=0}^{n-1} |t_\ell| H^\ell \ . \tag{3.1}$$

Then

$$|q_i^* - q_i| \le 2nH^{-i} \max_{0\le\ell<n} |r_\ell|/(H-1) \ , \qquad i = 0,1,\ldots,m-n \ . \tag{3.2}$$

Proof. The vector $[Q_i^*(H) - q_i H^i,\ i = 0,1,\ldots,m-n]$ is the image of the inverse DFT (K) applied to the vector

$$\left[\frac{r(H\omega^j)}{t(H\omega^j)} \ , \quad j = 0,1,\ldots,K-1\right] \ ,$$

and thus

$$\max_{0\le i<m-n} |Q_i^*(H) - q_i H^i| \le \sum_{j=0}^{K-1} \left|\frac{r(H\omega^j)}{t(H\omega^j)}\right| \le \sum_{j=0}^{K-1} |r(H\omega^j)| / \min_{0\le j<K} |t(H\omega^j)|$$

$$\le n\sum_{\ell=0}^{n-1} |r_\ell| H^\ell / \min_{0\le j<K} |t(H\omega^j)| \ .$$

Since $t_n = 1$, the assumption (3.1) implies that $min_{0\le j>K}|t(H\omega^j)| \ge 0.5H^n$. On the other hand, clearly,

$$\sum_{\ell=0}^{n-1} |r_\ell| H^\ell \le \left(\max_{0\le\ell<n} |r_\ell|\right) \sum_{\ell=0}^{n-1} H^\ell = \max_{0\le\ell<n} |r_\ell| \ \frac{H^n - 1}{H-1} \ .$$

Therefore,

$$\max_{0\le i\le m-n} |Q_i^*(H) - q_i H^i| \le 2n \max_{0\le\ell<n} |r_\ell|/(H-1) \ ,$$

and since $Q_i^*(H) = q_i^* H^i$, Proposition 3.1 follows.

□

Remark 3.1. [BP] provides us with the bound

$$\sum_{\ell=0}^{m-n} |q_\ell| \le (1+T)^{m-n} \sum_{i=n}^{m} |s_i| \ ,$$

c) the values $s(H\omega^j)/t(H\omega^j)$, $j = 0, 1, \ldots, K-1$ [by means of K concurrent divisions],

d) the coefficients of the polynomial

$$Q^*(H, y) = \sum_{i=0}^{m-n} Q_i^*(H) y^i$$

such that

$$Q^*(H, \omega^j) = \frac{s(H\omega^j)}{t(H\omega^j)} , \quad j = 0, 1, \ldots, K-1$$

[by means of the inverse DFT (K), for the cost bounded by c_A of (2.1)],

e) the coefficients of the polynomial

$$q^*(x) = \sum_{i=0}^{m-n} q_i^* x^i , \quad q_i^* = Q_i^*(H)/H^i$$

[by means of the parallel prefix algorithm, for the cost $O_A(\log(m-n), \frac{(m-n)}{\log(m-n)})$]; output $q_0^*, \ldots, q_{m-n}^*$, the computed approximations to $q_0, \ldots, q_{m-n}$.

Clearly, the overall asymptotic computational cost of this algorithm is still bounded by c_A of (2.1).

2. APPROXIMATE POLYNOMIAL DIVISION

We will need to apply DFT (K) for some fixed K exceeding $m-n$. We may choose $K = 2^k$ where $k = \lceil \log_2(m-n+1) \rceil$, and then we may perform DFT (K) by means of FFT.

Let hereafter $\omega = \omega_K$ denote a primitive $K-th$ root of 1.

First let $r(x)$ vanish identically in x; then the e.-i.t. immediately gives us $q(x)$.

Algorithm 1. Successively compute

a) $s(\omega^j)$ and $t(\omega^j)$ for $j = 0, 1, \dots, K-1$ [two forward DFTs (K)],

b) $s(\omega^j)/t(\omega^j)$ for $j = 0, 1, \dots, K-1$ [K concurrent divisions],

c) the coefficients $q_0, \dots, q_{m-n}$ of the polynomial $q(x) = s(x)/t(x)$ [an inverse DFT (K)].

The overall asymptotic complexity of these computations is bounded by

$$c_A = O_A\left(\log m, K + \frac{m}{\log(m/K)}\right) = O_A(\log m, m)\ . \tag{2.1}$$

Let us extend Algorithm 1 and the latter complexity estimates to approximate evaluation of $q(x)$ where $r(x)$ may not vanish. The underlying idea is that (1.1) implies that

$$q(x) = \frac{s(x)}{t(x)} - \frac{r(x)}{t(x)}$$

and the ratio $r(x)/t(x)$ converges to 0 as $|x| \to \infty$ since $\deg r(x) < \deg t(x)$. Thus, we will apply the e.-i.t. for large $|x|$ and will ignore the term $r(x)/t(x)$. Specifically, we will chose a large positive H (to be specified later on) and will proceed as follows:

Algorithm 2. Successively compute

a) the coefficients $S_g(H) = s_g H^g$ and $T_\ell(H) = t_\ell H^\ell$ of the polynomials $S(H, y) = s(Hy)$ and $T(H, y) = t(Hy)$ [by means of the parallel prefix algorithm ([EG], [KR]) for the cost $O_A(\log m, m/\log m)$],

b) the values $s(H\omega^j) = S(H, \omega^j)$ and $t(H\omega^j) = T(H, \omega^j)$, $j = 0, 1, \dots, K-1$ [by means of two DFTs (K), for the cost bounded by c_A of (2.1)],

It was another surprise for us to find out that the much more elementary e.-i.t. has led us to exactly the same computations as in [B82] and [S82], that is, all the three algorithms compute the same output and intermediate values.

Finally, we would like to mention that the exact evaluation of $q(x)$ [and $r(x)$] for the parallel cost $O_A(\log m, m)$ remains an open problem. The current records for the complexity of the exact evaluation of $q(x)$ are due to [BPa,b,c] and are given by $O_A(h \log m, (m/h)(1 + 2^{-h} \log^{(h)} m))$, for any h, where $1 \leq h \leq \log^* m$, $\log^{(0)} m = m$, $\log^{(h)} m = \log \log^{(h-1)} m$, $h = 1, \ldots, \log^* m$, $\log^* m = \min\{h : \log^{(h)} m \leq 1\}$.

1. INTRODUCTION

The *evaluation-interpolation technique (e.-i.t.)* (see [Toom 63], [AHU], [BM], [BPa]) is a customary powerful tool for several polynomial computations. In this note we will show its unusual application, to polynomial division with a remainder, that is, to the problem of computing the coefficients of two polynomials,

$$q(x) = \sum_{i=0}^{m-n} q_i x^i \quad \text{and} \quad r(x) = \sum_{\ell=0}^{n-1} r_\ell x^\ell \ ,$$

such that

$$s(x) = t(x)q(x) + r(x) \tag{1.1}$$

given two polynomials

$$s(x) = \sum_{g=0}^{m} s_g x^g \quad \text{and} \quad t(x) = \sum_{\ell=0}^{n} t_\ell x^\ell \ , \quad t_n = 1 \ .$$

Actually, we will just seek $q(x)$, which will immediately give us $r(x) = s(x) - t(x)q(x)$.

If $r(x)$ identically vanishes, the evaluation of $q(x)$ by means of the e.-i.t. is straightforward (see Algorithm 1 in Section 2) and supports the record asymptotic parallel arithmetic complexity estimates $O_A(\log m, m)$. Here and hereafter, we assume the customary arithmetic PRAM model of parallel computing ([EG], [KR]) and let $O_A(T, P)$ denote the simultaneous bounds $O(Th)$ and $O(P/h)$ for any h, $1 \le h \le P$, on the arithmetic time and the number of processors involved. [Actually, what we need is essentially any model of parallel computing that supports the bounds $O_A(\log K, K)$ on the complexity of the discrete Fourier transform at the $K - th$ roots of 1, which we will denote DFT (K).]

It is more surprising that we will be able to extend the same e.-i.t. to the general case, where $r(x)$ may not vanish, and to arrive at the same record complexity estimates $O_A(\log m, m)$ for any precision approximation to $q(x)$. These record estimates have been also supported by the algorithms of [B82] and [Sc82], based on quite sophisticated techniques of diagonalization of a certain matrix algebra and on Loran's power series expansion, respectively.

Approximate Polynomial Division with a Remainder by Means of Evaluation and Interpolation

Victor Pan[*),**),***),****)], **Elliot Landowne**[*)] **and Akimou Sadikou**[*)]

CUCS-030-90

Abstract

We apply the evaluation-interpolation techniques of A. Toom to approximate polynomial division with a non-vanishing remainder. Surprisingly, this elemetary technique immediately leads us to the same computations and to the same record parallel complexity estimates for this problem, $O_A(\log m, m)$, as do the far more sophisticated algorithms of D. Bini and A. Schönhage, and our approach enables us to simplify the error analysis too.

[*)]Ph.D. Program in Computer Science, Graduate School and University Center, CUNY, 33 West 42nd St., New York, NY 10036. (Supported by PSC CUNY Awards 668541 and 669210).

[**)]Computer Science Dept., Columbia University, New York, NY 10027.

[***)]Computer Science Dept., SUNYA, Albany, NY 12222. (Supported by NSF Grant CCR 8805782.)

[****)]Mathematics and Computer Science Dept., Lehman College, CUNY, Bronx, NY 10468.