FAST COMPUTATION OF CONTINUED FRACTIONS

Omer Eğecioğlu[1], Çetin K. Koç [2] and Josep Rifà i Coma[3]

[1]Department of Computer Science, University of California, Santa Barbara, CA 93106

[2]Department of Electrical Engineering University of Houston, Houston, TX 77204

[3]Dpt. d'Informàtica. Fac. Ciències, Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain

Abstract—We give an $O(\log n)$ algorithm to compute the nth convergent of a periodic continued fraction. The algorithm is based on matrix representation of continued fractions, due to Milne-Thomson. This approach also allows for the computation of first n convergents of a general continued fraction in $O(\log n)$ time using $O(\frac{n}{\log n})$ processors.

1. INTRODUCTION

A substitution scheme for the computation of continued fractions (CFs) was recently proposed in [1]. This algorithm requires $O(\log n)$ number of operations to compute the nth convergent of a periodic CF. In this note we show that a formulation due to Milne-Thomson (matrix-vector approach) for arbitrary CFs specializes to the periodic case with the same time complexity, and is also efficiently parallelizable for general CFs by using parallel prefix techniques.

Algorithms for the computation of CFs can be used to compute quadratic surds, Fibonacci numbers, and solutions of second order linear recurrences. Fibonacci numbers, for example, can be computed using the following identity

$$\begin{bmatrix} F_{n+1} \\ F_n \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n \begin{bmatrix} F_1 \\ F_0 \end{bmatrix}, \tag{1}$$

which can be derived from a continued fraction expansion. We can find F_n in $O(\log n)$ time by computing the nth power of the above 2×2 matrix using the well-known binary method (square and multiply method) [4].

The algorithm we propose uses the matrix representation of a general CF, due to L. M. Milne-Thomson (see section 5.3, pages 108-110 in [7]). Using the Milne-Thomson matrix representation, the computation of the kth convergent of a general CF for $k = 1, 2, \ldots n$ is shown to follow from the computation of prefix products of n matrices. The resulting parallel algorithm can be used to compute all convergents of a general CF in $O(\log n)$ parallel time using $O(\frac{n}{\log n})$ processors.

2. FAST COMPUTATION OF PERIODIC CONTINUED FRACTIONS

Consider the nth convergent

$$\frac{p_n}{q_n} = a_1 + \frac{b_2}{a_2+} \frac{b_3}{a_3+} \cdots \frac{b_n}{a_n} \tag{2}$$

of a given CF. It is shown by Milne-Thomson that p_n and q_n can be represented as the product of n 2×2 matrices

$$\begin{bmatrix} p_n & p_{n-1} \\ q_n & q_{n-1} \end{bmatrix} = \begin{bmatrix} a_1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_2 & 1 \\ b_2 & 0 \end{bmatrix} \begin{bmatrix} a_3 & 1 \\ b_3 & 0 \end{bmatrix} \cdots \begin{bmatrix} a_n & 1 \\ b_n & 0 \end{bmatrix}. \tag{3}$$

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TeX

The above representation provides an obvious alternative for the fast computation of the nth convergent of a CF. Let

$$A_k = \begin{bmatrix} a_k & 1 \\ b_k & 0 \end{bmatrix} \text{ for } 1 \le k \le n,$$

with $b_1 = 1$. We can then write (3) concisely as

$$\begin{bmatrix} p_n & p_{n-1} \\ q_n & q_{n-1} \end{bmatrix} = A_1 A_2 A_3 \cdots A_n.$$

A periodic CF with period j is a CF of the form

$$a_1 + \frac{b_2}{a_2+} \frac{b_3}{a_3+} \cdots \frac{b_i}{a_i+} \overline{\left(\frac{e_1}{f_1+} \frac{e_2}{f_2+} \cdots \frac{e_j}{f_j+} \right)}, \tag{4}$$

where i and j are constants and the bar denotes the periodic part. Define

$$B_k = \begin{bmatrix} e_k & 1 \\ f_k & 0 \end{bmatrix} \text{ for } 1 \le k \le j.$$

Let $G_k = A_1 A_2 \cdots A_k$ and $H_r = B_1 B_2 \cdots B_r$ for $1 \le k \le i$ and $1 \le r \le j$. If $n < i + j$, then the nth convergent $\frac{p_n}{q_n}$ can be computed from

$$\begin{bmatrix} p_n & p_{n-1} \\ q_n & q_{n-1} \end{bmatrix} = G_k, \text{ or } \begin{bmatrix} p_n & p_{n-1} \\ q_n & q_{n-1} \end{bmatrix} = G_i H_r \tag{5}$$

for $k < i$ and $r < j$. If $n \ge i + j$, then $n = i + sj + r$ with $s \ge 1$ and $0 \le r \le j - 1$. Thus

$$\begin{bmatrix} p_n & p_{n-1} \\ q_n & q_{n-1} \end{bmatrix} = G_i \overbrace{H_j H_j \cdots H_j}^{s \text{ times}} H_r = G_i H_j^s H_r \tag{6}$$

The above representation of a periodic CF immediately yields an $O(\log n)$ algorithm:

THEOREM 1. The nth convergent $\frac{p_n}{q_n}$ of a periodic continued fraction can be computed in $O(\log n)$ time.

Proof: We first compute G_i, H_r, and H_j using $O(i)+O(j)$ arithmetic operations. Note that H_r is computed during the computation of H_j since $r < j$. We then proceed to compute H_j^s in $O(\log s)$ arithmetic operations using the binary method [4]. The final matrix product $G_i H_j^s H_r$ requires only $O(1)$ arithmetic operations. Since i and j are constants, the total number of arithmetic operations is

$$O(\log s) = O(\log \frac{n-i-r}{j}) = O(\log n).$$

We see that the Milne-Thomson matrix representation provides an $O(\log n)$ algorithm for computing periodic CFs. For the justification of the identity (1) to compute the Fibonacci numbers, consider the well-known CF expansion of the golden ratio:

$$\frac{1+\sqrt{5}}{2} = 1 + \frac{1}{1+} \frac{1}{1+} \frac{1}{1+} \cdots,$$

in which the nth convergent is $\frac{F_{n+1}}{F_n}$. The Milne-Thomson construction gives

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \cdots \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n,$$

which is equivalent to (1).

3. PARALLEL COMPUTATION OF GENERAL CONTINUED FRACTIONS

The Milne-Thomson matrix representation can be utilized to design a parallel algorithm for the computation of a general CF as well.

THEOREM 2. All convergents $\frac{p_k}{q_k}$ for $k = 1, 2, 3 \ldots, n$ of a general CF can be computed computed in $O(\log n)$ parallel time using $O(\frac{n}{\log n})$ processors.

Proof: The computation of p_k and q_k for all $k = 1, 2, \ldots, n$ requires the computation of the prefix products of the matrices A_k for $1 \leq k \leq n$. This can be done in $O(\log n)$ matrix multiplications with $O(n)$ processors using a parallel prefix algorithm [6,3,2,5,8]. Since each matrix is 2×2, the total number of arithmetic operations is equal to $O(\log n)$. With a little more effort, we can reduce the required number of processors by a factor of $O(\log n)$: the prefixes of n elements can be computed in $O(\frac{n}{p} + \log p)$ time using p processors [5]. Choosing $p = \frac{n}{\log n}$, the parallel time becomes

$$O\left(\frac{n}{\frac{n}{\log n}} + \log \frac{n}{\log n}\right) = O(\log n - \log \log n) = O(\log n).$$

Thus the first n convergents of a general CF can be computed in $O(\log n)$ time with $O(\frac{n}{\log n})$ processors.

REFERENCES

1. K. L. Chung, W. C. Chen, and F. C. Lin, Fast Computation of Periodic Continued Fractions, *Information Processing Letters* **33**, 67–72 (1989).
2. F. E. Fich, New Bounds for Parallel Prefix Circuits, *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, 100–109 (1983).
3. A. G. Greenberg, R. E. Ladner, M. Paterson, and Z. Galil, Efficient Parallel Algorithms for Linear Recurrence Computation, *Information Processing Letters* **15**, 31–35 (1082).
4. D. E. Knuth, *The Art of Computer Programming Volume 2, Seminumerical Algorithms, 2nd Edition*, Addison-Wesley Publishing Company, (1981).
5. C. P. Kruskal, L. Rudolph, and M. Snir, The Power of Parallel Prefix, *IEEE Transactions on Computers* **34**, 965–968 (1985).
6. R. Ladner and M. Fischer, Parallel Prefix Computation, *Journal of ACM* **27**, 831–838 (1980).
7. L. M. Milne-Thomson, *The Calculus of Finite Differences, 2nd Unaltered Edition*, Chelsea Publishing Company, New York, (1981).
8. M. Snir, Depth-Size Trade-offs for Parallel Prefix Computation, *Journal of Algorithms* **7**, 185–201 (1986).