Information Processing Letters 33 (1989/90) 67–72
North-Holland

10 November 1989

FAST COMPUTATION OF PERIODIC CONTINUED FRACTIONS

Kuo-Liang CHUNG, Wen-Chin CHEN and Ferng-Ching LIN

Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan 10764, Taiwan

Communicated by David Gries
Received 1 June 1989
Revised 3 July 1989

Periodic continued fractions are useful for representing or approximating numbers. Based on a self-substitution concept, we derive an $O(\log n)$ algorithm to compute periodic continued fractions. Two application examples, the approximation of a quadratic surd number and the solving of second-order linear recurrence, are presented.

Keywords: Design of algorithms, periodic continued fraction, self-substitution, second-order linear recurrence, Fibonacci number

1. Introduction

The following form represents a continued fraction (CF):

$$V_n = a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{\cdots \cfrac{}{a_{n-1} + \cfrac{b_n}{a_n}}}} \tag{1}$$

The CFs provide a useful means for solving equations such as three-term recurrence and Recatti equation and for approximating numbers such as e, π, $\arctan(x)$ [2,6,7].

The above representation for CF is typographically cumbersome and can be replaced by the more compact form:

$$V_n = a_1 + \frac{b_2}{a_2+}\,\frac{b_3}{a_3+}\cdots\frac{b_n}{a_n}.$$

A CF with period p can be expressed as

$$\begin{aligned} P &= a_1 + \frac{b_2}{a_2+}\,\frac{b_3}{a_3+}\cdots\frac{b_k}{a_k+}\,\frac{e_1}{f_1+}\,\frac{e_2}{f_2+}\cdots\frac{e_p}{f_p+}\,\frac{e_1}{f_1+}\,\frac{e_2}{f_2+}\cdots\frac{e_p}{f_p+}\,\frac{e_1}{f_1+}\,\frac{e_2}{f_2+}\cdots \\ &= a_1 + \frac{b_2}{a_2+}\,\frac{b_3}{a_3+}\cdots\frac{b_k}{a_k+}\,\overline{\frac{e_1}{f_1+}\,\frac{e_2}{f_2+}\cdots\frac{e_p}{f_p+}} \end{aligned} \tag{2}$$

for some constant k.

In this paper, we first introduce a substitution scheme as the base for designing fast computations of CFs. Then we embed a self-substitution concept into the scheme to design an O(log n) algorithm for computing any periodic CF. Two examples, the approximation of $\sqrt{7}$ and the solving of second-order linear recurrence, are used to illustrate the application of our method.

2. The substitution scheme

Consider the problem of computing V_n in equation (1). At first glance, it seems that the rational form of V_n cannot be efficiently obtained because there exists a dependence relation in the stair-like, bottom-up computations of V_n. Fortunately, the dependence relation can be resolved by decomposing V_n into disjoint sub-CFs that can be processed separately. For simplicity, we use V_4 to illustrate this idea. V_4 can be decomposed into four sub-CFs:

$$V_4 = a_1 + X_3, \qquad X_3 = \frac{b_2}{a_2 + X_2}, \qquad X_2 = \frac{b_3}{a_3 + X_1}, \qquad X_1 = \frac{b_4}{a_4}. \tag{3}$$

Each of the right-hand sides of the equations in (3) is a sub-CF. The pseudo-variables X_3, X_2 and X_1 serve as the substitution bridges without any functional meaning.

Let the general rational form $(c_1 + c_2X_i)/(c_3 + c_4X_i)$ be represented by the notation $(c_1, c_2, c_3, c_4, X_i)$. The equations in (3) can be rewritten as:

$$V_4 = \frac{a_1 + X_3}{1} = (a_1, 1, 1, 0, X_3), \qquad X_3 = \frac{b_2}{a_2 + X_2} = (b_2, 0, a_2, 1, X_2),$$

$$X_2 = \frac{b_3}{a_3 + X_1} = (b_3, 0, a_3, 1, X_1), \qquad X_1 = \frac{b_4}{a_4 + X_0} = (b_4, 0, a_4, 1, X_0),$$

where the pseudo-variable X_0 will be replaced by zero after all the substitution operations are completed.

Since substituting $X_i = (d_1 + d_2X_j)/(d_3 + d_4X_j)$ into $(c_1 + c_2X_i)/(c_3 + c_4X_i)$ gives

$$\big((c_1d_3 + c_2d_1) + (c_1d_4 + c_2d_2)X_j\big)/\big((c_3d_3 + c_4d_1) + (c_3d_4 + c_4d_2)X_j\big),$$

a substitution operation $\circ$ with eight multiplications and four additions can be defined as

$$(c_1, c_2, c_3, c_4, X_i) \circ (d_1, d_2, d_3, d_4, X_j)$$
$$= (c_1d_3 + c_2d_1, c_1d_4 + c_2d_2, c_3d_3 + c_4d_1, c_3d_4 + c_4d_2, X_j).$$

Lemma 2.1. *Substitution operation $\circ$ is associative.*

Proof. For any $(c_1, c_2, c_3, c_4, X_i)$, $(d_1, d_2, d_3, d_4, X_j)$ and $(e_1, e_2, e_3, e_4, X_k)$, we have

$$\big[(c_1, c_2, c_3, c_4, X_i) \circ (d_1, d_2, d_3, d_4, X_j)\big] \circ (e_1, e_2, e_3, e_4, X_k)$$
$$= (c_1d_3 + c_2d_1, c_1d_4 + c_2d_2, c_3d_3 + c_4d_1, c_3d_4 + c_4d_2, X_j) \circ (e_1, e_2, e_3, e_4, X_k)$$
$$= (c_1d_3e_3 + c_2d_1e_3 + c_1d_4e_1 + c_2d_2e_1, c_1d_3e_4 + c_2d_1e_4 + c_1d_4e_2 + c_2d_2e_2,$$
$$c_3d_3e_3 + c_4d_1e_3 + c_3d_4e_1 + c_4d_2e_1, c_3d_3e_4 + c_4d_1e_4 + c_3d_4e_2 + c_4d_2e_2, X_k)$$

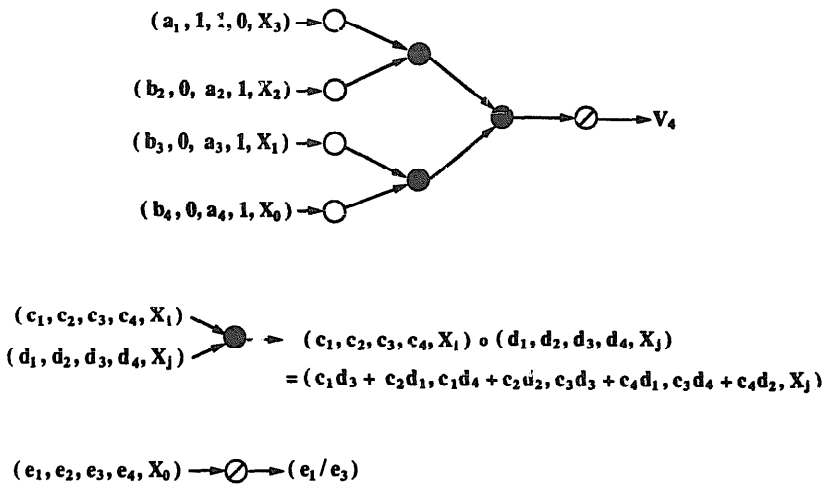

Fig. 1. A computation tree of V_4.

and

$$\begin{aligned}&(c_1, c_2, c_3, c_4, X_i)\circ\big[(d_1, d_2, d_3, d_4, X_j)\circ(e_1, e_2, e_3, e_4, X_k)\big]\\&=(c_1, c_2, c_3, c_4, X_i)\circ(d_1e_3+d_2e_1, d_1e_4+d_2e_2, d_3e_3+d_4e_1, d_3e_4+d_4e_2, X_k)\\&=(c_1d_3e_3+c_2d_1e_3+c_1d_4e_1+c_2d_2e_1, c_1d_3e_4+c_2d_1e_4+c_1d_4e_2+c_2d_2e_2,\\&\quad c_3d_3e_3+c_4d_1e_3+c_3d_4e_1+c_4d_2e_1, c_3d_3e_4+c_4d_1e_4+c_3d_4e_2+c_4d_2e_2, X_k).\end{aligned}$$

Both computation sequences give the same rational form. □

By Lemma 2.1, $V_4=(a_1, 1, 1, 0, X_3)\circ(b_2, 0, a_2, 1, X_2)\circ(b_3, 0, a_3, 1, X_1)\circ(b_4, 0, a_4, 1, X_0)$ can be computed in any order. Fig. 1 shows a computation tree for V_4. In Fig. 1, the white nodes at level 1 transmit data without changing them. The black nodes at levels 2 and 3 perform the function defined by substitution operation $\circ$. The slashed node at level 4 assigns zero to the pseudo-variable X_0 to obtain the desired result.

3. The algorithm

The periodic CF in (2) can be partitioned into the following sub-CFs:

$$\begin{aligned}P_n &= a_1 + \frac{b_2}{a_2+}\,\frac{b_3}{a_3+}\cdots\frac{b_k}{a_k+X_{\lfloor(n-k)/p\rfloor}},\\X_{\lfloor(n-k)/p\rfloor} &= \frac{e_1}{f_1+}\,\frac{e_2}{f_2+}\cdots\frac{e_p}{f_p+X_{\lfloor(n-k)/p\rfloor-1}},\\X_{\lfloor(n-k)/p\rfloor-1} &= \frac{e_1}{f_1+}\,\frac{e_2}{f_2+}\cdots\frac{e_p}{f_p+X_{\lfloor(n-k)/p\rfloor-2}},\\&\vdots\\X_1 &= \frac{e_1}{f_1+}\,\frac{e_2}{f_2+}\cdots\frac{e_p}{f_p+X_0'},\end{aligned}\tag{4}$$

where

$$X_0' = \begin{cases} X_0 & \text{if } (n-k)/p \text{ is an integer;} \\ \dfrac{e_1}{f_1+}\,\dfrac{e_2}{f_2+}\cdots\dfrac{e_j}{f_j+X_0} & \text{otherwise;} \end{cases}$$

and $j=(n-k)-p\lfloor(n-k)/p\rfloor$.

We first deal with the case when $(n-k)/p$ is an integer and a power of 2. It can be observed that all the sub-CFs in (4), except the first one, have exactly the same form. It is natural and economical to transform just one, say the last sub-CF, to its rational form and let the other inherit the same rational form associated with their own pseudo-variables implicitly. This arrangement of computations is legal since the substitution operation is associative by Lemma 2.1. Thus, (4) becomes

$$\begin{aligned} P_n &= a_1 + \frac{b_2}{a_2+}\,\frac{b_3}{a_3+}\cdots\frac{b_k}{a_k+X_{(n-k)/p}}, \\ X_{(n-k)/p} &= (c_1, c_2, c_3, c_4, X_{(n-k)/p-1}), \\ X_{(n-k)/p-1} &= (c_1, c_2, c_3, c_4, X_{(n-k)/p-2}), \\ &\vdots \\ X_1 &= (c_1, c_2, c_3, c_4, X_0), \end{aligned} \tag{5}$$

for some numbers c_1, c_2, c_3, and c_4. It takes only $O(p)$ time to complete this first stage of computation.

By a similar argument, it is then sufficient to perform the substitution operation on just one pair of the above rational forms, say $(c_1, c_2, c_3, c_4, X_1)$ and $(c_1, c_2, c_3, c_4, X_0)$. The other pairs inherit the same rational form, so their substitution operation becomes unnecessary and can be eliminated. After that, (5) becomes

$$\begin{aligned} P_n &= a_1 + \frac{b_2}{a_2+}\,\frac{b_3}{a_3+}\cdots\frac{b_k}{a_k+X_{(n-k)/p}}, \\ X_{(n-k)/p} &= (d_1, d_2, d_3, d_4, X_{(n-k)/p-2}), \\ X_{(n-k)/p-2} &= (d_1, d_2, d_3, d_4, X_{(n-k)/p-4}), \\ &\vdots \\ X_2 &= (d_1, d_2, d_3, d_4, X_0), \end{aligned}$$

for some numbers d_1, d_2, d_3, and d_4. This special type of substitution, the self-substitution, takes O(1) time. By applying self-substitution $\log((n-k)/p)$ times (the height of a computation tree), (5) is reduced to

$$\begin{aligned} P_n &= a_1 + \frac{b_2}{a_2+}\,\frac{b_3}{a_3+}\cdots\frac{b_k}{a_k+X_{(n-k)/p}}, \\ X_{(n-k)/p} &= (g_1, g_2, g_3, g_4, X_0) = \frac{g_1}{g_3}, \end{aligned}$$

for some numbers g_1, g_2, g_3, and g_4.

It takes $O(k)$ time in the final stage to obtain the rational form of P_n. Totally, $O(p) + O(\log(n-k)/p) + O(k) = O(\log n)$ time is required to complete the computation. It is also clear that only O(1) space is needed in the whole process.

Next we discuss the second case when $(n-k)/p$ is an integer but not a power of 2. Let the binary representation of $(n-k)/p$ be $b(q)\dots b(1)b(0)$, where $b(q)=1$ is the most significant bit. That is, $(n-k)/p=\sum_{i=0}^{q} b(i)2^i$. For convenience, we use a Π-notation to represent substitution sequences. $X_{(n-k)/p}$ in (5) can be expressed as

$$X_{(n-k)/p}=\left(\prod_{\substack{i=q\\ b(i)=1}}^{0}(c_1, c_2, c_3, c_4)^{2^i}, X_0\right).$$

For example, if $(n-k)/p=11$ then $b(3)b(2)b(1)b(0)=1011$. Therefore, we have

$$X_{11}=\left((c_1, c_2, c_3, c_4)^8\circ(c_1, c_2, c_3, c_4)^2\circ(c_1, c_2, c_3, c_4)^1, X_0\right).$$

According to the self-substitution concept and the information $b(q)\dots b(1)b(0)$, the rational form of $X_{(n-k)/p}$ can be obtained in $O(\lceil\log(n-k)/p\rceil)$ time using O(1) space.

At last, we have the case when $(n-k)/p$ is not an integer. By the arguments in the first and second cases, after $O(p)+O(\lceil\log(n-k)/p\rceil)$ time has passed, (4) becomes

$$P_n=a_1+\frac{b_2}{a_2+}\frac{b_3}{a_3+}\cdots\frac{b_k}{a_k+X_{\lfloor(n-k)/p\rfloor}},$$

$$X_{\lfloor(n-k)/p\rfloor}=\left(\prod_{\substack{i=q\\ b(i)=1}}^{0}(c_1, c_2, c_3, c_4)^{2^i}, X_0'\right)=(h_1, h_2, h_3, h_4, X_0'),$$

$$X_0'=\frac{e_1}{f_1+}\frac{e_2}{f_2+}\cdots\frac{e_j}{f_j+X_0}.$$

The following main theorem summarizes the above results.

Theorem 3.1. *Any periodic CF can be computed in* O(log n) *time with* O(1) *space.*

4. Application examples

In this section, we give examples to show the power of the method described in Section 3. As the first example, the special quadratic surd number $\sqrt{7}$ can first be converted into the periodic CF $[2, 1, 1, 1, 4, 1, 1, 1, 4, \dots]=[2, \overline{1, 1, 1, 4}]$ [6]. Then using our algorithm, a good approximation to $\sqrt{7}$ can be obtained in O(log n) time.

The second example is to solve the second-order linear recurrence $x_n=ax_{n-1}+bx_{n-2}$ with given coefficients a, b and initial values x_0, x_1. By using the CF expansion of the ratio x_n/x_{n-1} and the substitution scheme, we get

$$\frac{x_n}{x_{n-1}}=a+\frac{b}{a+}\frac{b}{a+}\cdots\frac{b}{a+}\frac{bx_0}{x_1}=a+Y,$$

$$Y=\frac{b}{a+}\frac{b}{a+}\cdots\frac{b}{a+Z},\qquad Z=\frac{bx_0}{x_1+X_0}.$$

The rational form of Y can be computed in O(log n) time. Then, it takes O(1) time to obtain the rational form of x_n/x_{n-1} and hence its numerator x_n.

Corollary 4.1. *The second-order linear recurrence* $x_n = ax_{n-1} + bx_{n-2}$ *can be solved in* O(log n) *time.*

The Fibonacci number F_n is defined by

$$F_0 = 0, \qquad F_1 = 1, \qquad F_n = F_{n-1} + F_{n-2} \quad \text{for } n \geq 2.$$

The recurrence formula of the Fibonacci number is only a special case of the second-order linear recurrence where $a = 1$, $b = 1$, $x_0 = 0$ and $x_1 = 1$. (One can easily verify that F_{n+1}/F_n equals the nth convergent of the simple continued fraction $[\overline{1}]$. We therefore have the following corollary.

Corollary 4.2. *The Fibonacci number* F_n *can be computed in* O(log n) *time.*

5. Concluding remarks

In this paper, a new substitution scheme is proposed to compute the CFs efficiently. We then embed a self-substitution concept into the scheme to design a general O(log n) algorithm for computing any periodic CF. In the past, many researchers [3,4,5,8] have presented O(log n) algorithms for computing F_n based on the matrices–vector approach:

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} F_1 \\ F_0 \end{pmatrix}.$$

However, F_n is just one instance of periodic CFs and most CFs cannot be computed using the matrices–vector approach.

Based on our substitution scheme, we have constructed a new parallel computation model for computing general CFs [1]. This model allows all prefix values of a CF to be computed in O(log n) time on a cost-optimal network in which $O(n/\log n)$ processors communicate via an inverse perfect shuffle routing mechanism.

References

[1] K.L. Chung, F.C. Lin and C.W. Chen, Parallel computation of continued fractions, to be submitted.

[2] L. Euler, De fractionibus continuis dissertatio, Personal manuscript, 1737 (An essay on continued fractions, translated by M.F. Wyman and B.F. Wyman, *Math. Systems Theory* **18** (1985) 295–328).

[3] D. Gries and G. Levin, Computing Fibonacci numbers (and similarly defined functions) in log time, *Inform. Process. Lett.* **11** (2) (1980) 68–69.

[4] A.J. Martin and M. Rem, A representation of the Fibonacci algorithm, *Inform. Process. Lett.* **19** (2) (1984) 67–68.

[5] F.J. Urbanek, An O(log n) algorithm for computing the nth element of the solution of a difference equation, *Inform. Process. Lett.* **11** (2) (1980) 66–67.

[6] H.S. Wall, *Analytic Theory of Continued Fractions* (Van Nostrand, New York, 1948).

[7] J.B. William and W.J. Thron, *Continued Fractions: Analytic Theory and Applications* (Addison-Wesley, Reading, MA, 1980).

[8] T.C. Wilson and J. Shortt, An O(log n) algorithm for computing general order-k Fibonacci numbers, *Inform. Process. Lett.* **10** (2) (1980) 68–75.