J. Austral. Math. Soc. **21** (Series B) (1979), 84–89

SERIES OPERATIONS USING MINIMUM STORAGE

A. N. STOKES

(Received 22 May 1978)

(Revised 25 October 1978)

Abstract

An algorithm is given for transforming a polynomial with n coefficients to a continued fraction accurate to the same order. Only n numbers are held in storage at each stage. An extension to produce an inverse polynomial, also accurate to order n, is described.

This paper describes an algorithm for forming a continued fraction from a power series, and a consequent method of inverting the series. There are existing procedures for performing such tasks: continued fraction coefficients are usually found by the Viskovatoff algorithm [3] or the quotient-difference algorithm [1], while inversion can be performed by solving the sequence of equations formed when the series and its inverse are multiplied and the higher order coefficients set equal to zero [4].

In all these methods, to deal with n coefficients requires the retention in storage of about $2n$ numbers. The procedures described here need only storage space for n numbers. This is an important gain where the algorithms are used on small calculators. The arithmetic required for continued fraction formation is similar to that of alternative methods, although inversion requires about twice as many operations.

DEFINITIONS. *Let $\mathscr{C}$ be a contour in the complex plane terminating at $x = 0$.*

A sequence $\{r_i;\ i = 0,\ldots,n-1\}$ containing no zeroes is said to be a continued fraction expansion (c.f.e.) of order n of a function $f(x)$ if the sequence of remainder

functions $\{R_i(x), i = 0, \ldots, n\}$ *defined by*

$$R_0(x) = f(x),$$

$$R_{i+1}(x) = (r_i/R_i(x) - 1)/x, \quad i = 0, \ldots, n-1, \tag{1}$$

has the property that each $R_i(x)$ *is bounded as* $x \to 0$ *along* $\mathscr{C}$.

This property is ensured if $R_n(x)$ is bounded, since then each $R_i(x) \to r_i$ as $x \to 0$ along $\mathscr{C}$, $i = 0, \ldots, n-1$.

The function $f(x)$ can, by applying (1), be written in the more familiar form

$$f(x) = \cfrac{r_0}{1 + \cfrac{xr_1}{1 + \ldots + \cfrac{xr_i}{1 + xR_{i+1}}}}. \qquad 0 \leqslant i \leqslant n-1$$

As a trivial but useful extension, the empty sequence is defined as a c.f.e. of order zero for $f(x)$, provided $f(x)$ is bounded as $x \to 0$ along $\mathscr{C}$.

The basic step

Suppose $f(x)$ has the nth-order c.f.e. $\{r_i\}$ defined above. Then we find for

$$g(x) = \frac{s_0 + u_0 x f(x)}{1 + v_0 x + w_0 x f(x)}, \quad s_0 \neq 0, \tag{2}$$

a c.f.e. of order $n+1$, by sequential construction of the remainder functions. Let $S_0(x) = g(x)$, and suppose that the ith remainder S_i has been found and expressed in the form

$$S_i = \frac{s_i + u_i x R_i}{1 + v_i x + w_i x R_i}, \quad \text{where } s_i \neq 0 \text{ and } 0 \leqslant i \leqslant n. \tag{3}$$

Then

$$S_i = \cfrac{s_i}{1 + \cfrac{v_i x + w_i x R_i}{1 + u_i x R_i / s_i}}$$

$$= \frac{s_i}{1 + xS_{i+1}} \tag{4}$$

where

$$S_{i-1} = \frac{v_i + w_i R_i - u_i R_i / s_i}{1 + u_i x R_i / s_i} \tag{5}$$

Since R_i is bounded as $x \to 0$ along $\mathscr{C}$, so is S_{i+1}, and s_i can be defined to be the *ith* coefficient of a c.f.e. for $g(x)$.

https://doi.org/10.1017/S0334270000001946 Published online by Cambridge University Press

If $i<n$ then (1) can be used to express R_i in terms of R_{i+1}, x and r_i, giving

$$S_{i+1} = \frac{s_{i+1}+u_{i+1}xR_{i+1}}{1+v_{i+1}x+xw_{i+1}R_{i+1}} \tag{6}$$

where

$$v_{i+1} = u_i v_i / s_i, \tag{7a}$$

$$u_{i+1} = v_i, \tag{7b}$$

$$s_{i+1} = v_i + w_i r_i - v_{i+1}, \tag{7c}$$

$$w_{i+1} = 1. \tag{7d}$$

We now proceed by induction from $i = 0$. Now S_0 is in the standard form (3), and is bounded as $x \to 0$ along $\mathscr{C}$, so s_0 is the first coefficient of the c.f.e., and S_1 as defined by (5) is the next remainder. If $n>0$, S_1 is converted to the form (6), with coefficients given by (7) and the process repeated. Termination occurs either when (7a) requires division by a zero coefficient s_i, or when S_{n+1} has been defined by (5). In the latter case, it is easy to verify that S_{n+1} is bounded as $x \to 0$ along $\mathscr{C}$.

An algorithm for converting a series to a continued fraction

Suppose the function $f(x)$ has a series expansion

$$f(x) = p_0 + p_1 x + \ldots + p_{n-1}x^{n-1} + x^n P_n(x), \tag{8}$$

where $P_n(x)$ is bounded as $x \to 0$ along $\mathscr{C}$, and it is required to find a c.f.e. of order n for $f(x)$.

In the evaluation of $f(x)$ by Horner's method, a sequence $Q_k(x)$ is defined by

$$Q_n(x) = P_n(x),$$

$$Q_{n-k}(x) = p_{n-k} + xQ_{n-k+1}(x), \quad k = 1, 2, \ldots, n. \tag{9}$$

Then

$$Q_0(x) = f(x).$$

Suppose that the function Q_{n-k+1} has a c.f.e. $\{q_{i,n-k+1};\ i = 0, \ldots, k-2\}$. Then the procedure of the previous section can be used to construct a c.f.e. $\{q_{i,n-k};\ i = 0, \ldots, k-1\}$ for Q_{n-k}, since (9) is a special case of (3). But, trivially, Q_n has a c.f.e. of order zero. Then a c.f.e. of order 1 is found for Q_{n-1}, and so on, until a c.f.e. of order n has been found for $Q_0 = f(x)$.

At any stage this process could be terminated by the occurrence of a zero coefficient. More will be said about this later.

https://doi.org/10.1017/S0334270000001946 Published online by Cambridge University Press

The complete algorithm can be set out as follows:

I. Set $k = n-2$, $q_{0,n-1} = p_{n-1}$.

II. Set $q_{0,k} = p_k$, $u_0 = 1$, $v_0 = 0$, $w_0 = 0$.

III. For $i = 0, 1, \ldots, n-k-1$, compute the terms

$$v_{i+1} = u_i q_{i,k+i}/q_{i,k}, \tag{10a}$$

$$u_{i+1} = v_i, \tag{10b}$$

$$q_{i+1,k} = v_i + w_i q_{i,k+1} - v_{i+1}, \tag{10c}$$

$$w_{i+1} = 1. \tag{10d}$$

IV. If $k > 0$, decrease k by 1 and return to step II. If $k = 0$, terminate.

Note that $v_{2i} = u_{2i+1} = 0$ $(i = 0, 1, 2, \ldots)$. Consequently the operations of (10a) are performed only if i is even. Also, since w_i is either 0 or 1, no multiplication is performed in (10c). In total the algorithm requires $n^2/4$ multiplications if n is even, or $(n^2-1)/4$ if n is odd, and the same number of divisions.

It is necessary to store each coefficient $q_{i,k+1}$ only until $q_{i+1,k}$ is computed by (10c). At this stage the following numbers are stored:

$$q_{j,k+1}, \quad j = i, \ldots, n-k-1,$$

$$q_{j,k}, \quad j = 0, \ldots, i,$$

$$p_j, \quad j = 0, \ldots, k-1.$$

This makes a total of n numbers. In addition either v_{i+1}, if i is even, or u_{i+1}, if i is odd, will be stored.

Treatment of zero and near-zero coefficients

The algorithm cannot proceed if any coefficient $q_{i,k}$ appearing in the denominator of (10a) is zero, when i is even. Numerical difficulties may arise if the denominator is small. In most cases such problems can be avoided by first multiplying the series (8) by $1+\varepsilon x$, for suitable ε. The algorithm is employed as far as finding the c.f.e. of $Q_1(x)$, then in the final stage the initial condition $v_0 = 0$ is replaced by $v_0 = \varepsilon$. This effects a division by $(1+\varepsilon x)$.

Certain patterns of zeroes cannot be fully eliminated by this procedure. For example, if three consecutive coefficients p_i were zero, one zero would remain. To remove this, the polynomial can be multiplied by another factor $1+\varepsilon_1 x$ which is then removed by setting $v_0 = \varepsilon_1$ when the c.f.e. of $Q_1(x)$ is formed.

There are more direct ways of dealing with zeroes as they are encountered, but they are complicated to program, and do not deal with near-zeroes.

https://doi.org/10.1017/S0334270000001946 Published online by Cambridge University Press

Inversion of series

The algorithm described can be used in reverse to convert a c.f.e. into a series expansion, again requiring storage of n numbers. So if a series is converted to a c.f.e., which is then inverted (which is trivial) and the inverted expression expanded as a series, the reciprocal series of $f(x)$ is obtained. The process uses twice the arithmetic but half the memory required for the usual algorithm described, for example, in [4].

Numerical properties and comparisons with other algorithms

Compared with the usual methods, the quotient-difference (q-d) algorithm and the Viskovatoff algorithm, the present algorithm forms a c.f.e. with about the same amount of arithmetic, using about half the memory. It is similar to the q-d algorithm, and could be derived from it; the derivation given here is direct, and gives an interpretation to the coefficients allowing zero values to be treated. These are a difficult problem in the q-d algorithm.

All the algorithms mentioned tend to be unstable when the ratios of series coefficients form a regular progression. This occurs because the continued fraction coefficients involve higher-order differences of the ratios. The q-d algorithm can be stabilized by forming these differences accurately in advance, and operating on them in such a way that accuracy is not lost [5]; other algorithms do not seem amenable to this treatment.

Henrici [2] has developed an ingenious algorithm for obtaining continued fraction coefficients using even less memory than the algorithm given here. The saving in memory is transferred to the operator, who must insert the required series coefficients before the calculation of each continued fraction coefficient. Much more numerical work is required.

An implementation on a programmable calculator

The following program implements the algorithm on a Texas Instruments SR 52 programmable calculator, which has 20 memory locations. To convert a polynomial $P(x) = p_0 + p_1 x + \ldots p_N x^N$ $(N \leqslant 19)$ press A and enter in sequence the ratios $p_N/p_{N-1}, \ldots, p_1/p_0$, pressing RUN after each entry. All coefficients must be non-zero. Denoting the coefficients of the c.f.e. of $P(x)$ by q_i $(i = 0, \ldots, N)$, then $q_0 = p_0$, and for $i = 1, \ldots, N$, each q_i appears in memory $19 - i$. If $N = 19$, then q_N appears in the display register.

https://doi.org/10.1017/S0334270000001946 Published online by Cambridge University Press

The program is:

0	IND	18	$1/x$	36	A	54	DSZ
1	EXC	19	−	37	CMS	55	D
2	0	20	RSET	38	LBL	56	HLT
3	0	21	LBL	39	E	57	LBL
4	−	22	B	40	1	58	C
5	SUB	23	1	41	9	59	1
6	B	24	INV	42	STO	60	9
7	=	25	SUM	43	0	61	STO
8	±	26	0	44	0	62	0
9	÷	27	0	45	CLR	63	0
10	IND	28	IND	46	LBL	64	HLT
11	EXC	29	RCL	47	D	65	±
12	0	30	0	48	IND	66	−
13	0	31	0	49	EXC	67	RSET
14	÷	32	IF ZERO	50	0		
15	SUB	33	E	51	0		
16	B	34	RTN	52	IF ZERO		
17	=	35	LBL	53	C		

References

[1] P. Henrici, "The quotient difference algorithm', in *Mathematical methods for digital computers* (ed. A. Ralston and H. S. Wilf) (New York: Wiley, 1967), vol. 2.

[2] P. Henrici, *Computational analysis with the HP-25 pocket calculator* (New York: Wiley, 1977).

[3] A. N. Khovanskii, *The application of continued fractions and their generalizations to problems in approximation theory* (Groningen: Noordhoff, 1963).

[4] D. E. Knuth, *The art of computer programming. Vol. 2. Seminumerical algorithms* (Reading, Mass.: Addison-Wesley, 1969).

[5] A. N. Stokes, "A stable quotient-difference algorithm" (to appear).

C.S.I.R.O. Division of Mathematics and Statistics
P.O. Box 310
South Melbourne
Victoria, 3025

https://doi.org/10.1017/S0334270000001946 Published online by Cambridge University Press