TR-88-011

A FAST ALGORITHM FOR THE SUMMATION OF SERIES

TR-88-011

A Fast Algorithm for the Summation of Series

Ning–Yang Baby Wang* and Ruei–Chuan Chang**

* Ning–Yang Baby Wang is with the Institute of Information Science, Academia Sinica, Nankang, Taipei, Taiwan, Republic of China.

** Ruei–Chuan Chang is with the Institute of Computer Engineering, National Chiao Tung University, Hsinchu, Taiwan and Institute of Information Science, Academia Sinica, Nankang, Taipei, Taiwan, Republic of China.

Correspondence Address:

Professor Ruei–Chuan Chang
Institute of Information Science
Academia Sinica
Nankang, Taipei, Taiwan
Republic of China 11529

A fast algorithm for computing the summation of series is derived. If the error tolerence is given, the algorithm will try every time in which the numbef of terms is twice as much as the one of the previous time until the error is tolerable. The required number of operations, which is much lesser than that of the previous result, is linearly propotional to the number of terms in the partial sum of the original series we wish to evaluate. That is, we can use the same amount of effort to get more precious result.

1. Introduction:

In the previous papers Longman[2], Longman[3] and Lepora and Gabutti[1], some methods were provided for accelerating the numerical convergence of finite series. In [1], a recursive algorithm was used to implement the summation of the following series

$$s(x) = \mu_1 - \mu_2 x + \mu_3 x^2 - \mu_4 x^3 + \cdots. \qquad \cdots\cdots\cdots\cdots\cdots\cdots (1)$$

Let

$$S_m \equiv S_m(x) = \sum_{k=1}^{m} (-x)^{k-1} \mu_k \,, \; m = 1, 2, \cdots, \qquad \cdots\cdots\cdots\cdots\cdots\cdots (2)$$

be the partial sum of the series $s(x)$. The mentioned transformation in [3] consists in replacing S_m by

$$\overline{S}_m \equiv \overline{S}_m(x) = \sum_{k=1}^{2m} \alpha_k^{(m)} (-x)^{k-1} \mu_k \,, \; m = 1, 2, \cdots, \qquad \cdots\cdots\cdots\cdots\cdots (3)$$

where

$$\alpha_k^{(m)} \equiv \alpha_k^{(m)}(x) = (1+x)^{-m} \sum_{r=0}^{2m-k} \binom{m}{r} x^r \,, \qquad \cdots\cdots\cdots\cdots\cdots\cdots (4)$$

and

$$x \neq -1.$$

When x is positive the great advantage of $\overline{S}_m(x)$ over $S_m(x)$ is evident. Moreover, if x satisfies $0 < x \leq 1$ a further advantage is obtained. In [1], the algorithm is as follows. Let

$$W_{m,n+1} = (1+x)^{-1}(W_{m+2,n} + x \cdot W_{m+1,n}) \,, \quad \begin{matrix} n=0,1,\cdots,m \\ m=0,1,2,\cdots \end{matrix} \qquad \cdots\cdots\cdots\cdots (5)$$

where $x \neq -1$. Let S_m be given by (2), and let

$$W_{m,0} = S_m \,, \; m = 1, 2, \cdots. \qquad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (6)$$

Then

$$W_{0,n} = \overline{S}_n \,, \; n = 1, 2, \cdots, \qquad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (7)$$

where $\overline{S}_n$ is given by (3) and (4) with m replacing by n.

If it is required to compute $W_{0,1}$, $W_{0,2}$, $\cdots$, $W_{0,n}$ to make the error tolerable

eventually, the algorithm runs $O(n^2)$ steps. We hereby give a faster $(O(n))$ algorithm to evaluate the summation that satisfies the same error tolérance.

This paper is organized as follows. In section 2 we give the fast algorithm that implements the summation. The analysis of the algorithm is in section 3. Finally, we give the concludinge remarks in section 4.

2. The algorithms:

We are given the original series and the tolerance of error. First we compute $\overline{S}_2(x)$ and see if the corresponding error is tolerable. If so, the algorithm stops; otherwise we advance to compute $\overline{S}_4$ and see if the corresponding error is tolerable, and so forth. If it should evaluate $W_{0,n}$ to satisfies the given error tolerance in the algorithm of [1], $\overline{S}_{n^*}$, which algorithm A computes eventually, has the property $n^* < 2 \cdot n$.

[Algorithm A]:

Input: 1. the original series in (1)

2. the error tolerance ϵ

Output: a partial sum of the original series that satisfies the given error tolerance

Begin

Step 1. Let m, the number of terms of the partial sum, be 1.

Step 2. Set error = ∞.

Step 3. While error $> \epsilon$ do

Begin

let $m = m \cdot 2$;

call algorithm B to evaluate partial sum $\overline{S}_m(x)$;

compute the corresponding error;

End.

Step 4. Return (the partial sum we get eventually).

End.

If we are given the original series and m, the number of terms of the partial sum we wish to evaluate, algorithm B will compute $\overline{S}_m(x)$ directly from (3) and (4) rather than from (5), (6) and (7).

[Algorithm B]:

Input: 1. the original series in (1)

2. m, the number of terms of the partial sum in (2)

Output: the partial sum $\overline{S}_m(x)$ in (3) and (4)

Begin

Step 1. Compute $(-x)^0, (-x)^1, \cdots, (-x)^m$ and store them.

Step 2. Compute $\binom{m}{0}, \binom{m}{1}x, \binom{m}{2}x^2, \cdots, \binom{m}{m}x^m$ and $(1+x)^m$ and store them.

Step 3. Compute $Q_1 = \sum_{k=1}^{m} (-x)^{k-1}\mu_k$.

Step 4. Let $Q_2 = 0$ and $\alpha = (1+x)^m$.

Step 5. For i=1 while $i \leq m$ do

Begin

let $\alpha = \alpha - \binom{m}{m-i+1}x^{m-i+1}$;

let $Q_2 = Q_2 + \alpha \cdot (-x)^i \cdot \mu_{i+m}$;

end.

Step 6. Let $Q_2 = Q_2 \cdot \frac{(-x)^m}{(1+x)^m}$.

Step 7. Return $(Q_1 + Q_2)$.

End.

3. Analysis of the algorithms:

In algorithm B because step 3, 4, 6 and 7 need O(1) operations while step 1, 2 and 5 need O(m) operations if number of terms of the partial sum we want to evaluate is m, it runs in time O(m) clearly. If it is required to evaluate $W_{0,n}$ to make the error tolerable for the algorithm in [1], algorithm A needs $O(2)+O(4)+O(8)+\cdots+O(2^{\lfloor \log_2 n \rfloor + 1}) = O(n)$ steps, which is much better than $O(n^2)$ steps in algorithm in [1]. For the space complexity, in step 1 and 2 the terms $(-x)^0, (-x)^1, \cdots, (-x)^m$ and $\binom{m}{0}, \binom{m}{1}x, \binom{m}{2}x^2, \cdots, \binom{m}{m}x^m$ should be evaluated step by step and stored in order to be completed in O(m) time; therefore the space complexity is O(n).

4. Concluding remarks:

The algorithm we used is faster than the previous ones. However, some shorthands are in it. First of all, it needs to evaluate $\binom{m}{0}, \binom{m}{1}, \cdots, \binom{m}{m}$, which are quite large when m increases. And, if x is far away from 1, the term $(-x)^m$ is either extreme large or small when m is sufficiently large. Thus these might influence the error relative to this series.

Reference:

[1] Paolo Lepora and Bruno Gabutti, An Algorithm for the Summation of Series, Appl. Numer. Math. 3 (1987) 523–528.

[2] I.M. Longman, The summation of power series and Fourier series, J. of Comput. Appl. Math. 12&13 (1985) 447–457.

[3] I.M. Longman, the summation of series, Appl. Numer. Math. 2 (1986) 135–141.