Research Journal of Information Technology 3(2): 68-71, 2011
ISSN: 2041-3114

Submitted: March 21, 2011 Accepted: April 30, 2011 Published: September 30, 2011

Approximation of Mathematical Functions Using Horner's Scheme and Continued Fraction

A.A. Eludire
Department of Computer Science, Joseph Ayo Babalola University,
Ikeji Arakeji, Osun State, Nigeria

Abstract: The choice of optimal algorithms for fast computation of functions are of great interest in numerical theory and have increasing importance in the organisation and development of computer systems. The main problem here is the definition of such computational methods of universal character in use and such functional property that justifies the hardware implementation. This study examines the process of solving mathematical functions on computers using Horner's scheme and continued fraction. It concludes that procedures combining both algorithms may be efficiently used as an option in software for the approximation of linear systems of equations, serial and pipelined functions.

Key words: Approximation, continued fraction, Horner's scheme, iterative functions, pipelined functions, serial functions

INTRODUCTION

In digital signal processing and computer techniques there are problems dealing with trigonometric functions, transformation of coordinates, computation of elementary functions like e^x, lnx, $\sqrt{x}, \sqrt{x^2 + y^2}$, etc.

The solution of these types of functions on universal computers or specialised processor generally require considerable computer time and in some cases cannot even be realised. This is due to the fact that the Arithmetic-logical Unit (ALU) of digital computer is only capable of carrying out four main arithmetic operations and elementary logical operations and in special cases some specially defined operations, for example, scalar product (Al-Kurdi *et al*., 2002).

In most cases the solution of elementary function, finding of max and min, sum of whole part are realised on computers with the use of standard functions or universal subroutine. There are lots of standard functions for the solution of mathematical functions among which are:

- Exponential, polynomial, rational resolutions
- Iterative process
- Continued fraction

The last two are widely used lately, but the choice of standard function depends on many important characteristics of computer among which are:

- speed
- word length/word size
- form of number representation
- memory capacity etc.

For computers working on arbitrary word length and problem-oriented algorithms of iterative process are widely used. In connection with the need for reduction of interval between the implementation of MULTIPLY and DIVIDE operations and correspondingly increase the computational speed of functions wide use has been made of algorithms based on rational approximations, for example, the algorithm of resolving into continued fraction and Horner's scheme (Golub and van-Loan, 1989).

Wilkinson (1948) used the technique of iterative refinement for improving approximation of mathematical functions in a computer program during the design and building of the ACE computer at the National Physical Laboratory (Wilkinson, 1948). Iterative refinement has achieved wide use ever since and is exploited by most of the linear system expert drivers. Further applications of this technique are outlined in (Wheeler, 2002; Allenby and Redfern, 1989) and references therein.

The main objective of this study is to improve the process of approximating mathematical functions by using methods with proven level of stable results while reducing round off error and increasing the speed of polynomial computation.

APPROXIMATION METHODS OF FUNCTIONS

Approximation of functions using Horner's scheme: One of the most widely applied methods for polynomial computation is Horner's scheme. This is due to a number of reasons amongst which are:

- simplicity
- increased computational speed of polynomials
- reduction of round-off errors

Assuming that it is necessary to compute the following polynomial:

$$P_n(x) = \sum_{k=0}^{n} a_k x^k = a_0 + a_1 x_1 + \ldots + a_k x_k \quad (1)$$

instead of raising x to power *k* and multiplying by coefficient , (k = 0,...,n), Eq. (1) can be written as:

$$P_n(x) = ((\ldots((a_k x + a_{kl})x + a_{k2})x + \ldots)x + a_0 \quad (2)$$

In this case the problem has been brought to that of recurrent formula:

$$V_n = a_k;\ V_k = a_k + xV_{k+1},$$
$$(k = n\text{-}1,\ldots, 0) \text{ and } P_k = P_0 \quad (3)$$

In the direct computation of (1) it is necessary to carry out *n* multiplications and (n+1) n/2 additions if all rising to power is got by serial multiplication of *x*. But using Homer's scheme in (2) it is only *n* multiplication and *n* additions and if *m* coefficients shall be equal to 0 then the quantity of addition is reduced by m.

Approximation of functions using continued fraction: Continued fraction is one important source of obtaining rational function where the approximation of fraction can be obtained on one side and a reduced number of operations in the computation of elementary functions is obtained on the other. Continued, or continuous fraction is defined as expression:

$$b_0 = a_1/b_1 + a_2/b_2 + a_3/b_3 + \ldots\ldots$$

where,
a_k/b_k : k^{th} element of continued fraction
b_0 : first element
a_k : partial numerator
b_k : partial denominator

Different forms of writing continued fraction exists:

$$b_0 + a_1 / b_1 + a_2 / b_2 + \ldots a_k / b_k \quad (5)$$

or as used in practice:

$$b_0 + \sum \frac{a_n}{b_n} \quad (6)$$

Continued fraction limited by *n*-th element as in (6) is referred to as *n*-approaching continued fraction (4) written as:

$$f_n(x) = \frac{P_n}{Q_n} = b_0 + \frac{a_1}{b_1} + \frac{a_2}{b_2} + \ldots + \frac{a_n}{b_n} \quad (7)$$

or

$$f_n(x) = \frac{P_n(x)}{Q_n(x)} = \frac{b_0}{1} + \frac{a_1}{b_1} + \frac{a_2}{b_2} + \ldots + \frac{a_n}{b_n} \quad (8)$$

n-approaching fraction P_n/Q_n converging continued fraction type (4) approximate the unknown quantity towards the lower limit when even and upper when odd. As such *n*-approaching even-power forms increasing series while odd-power forms descending series.

Analyzing all the possible methods for implementing continued fraction, it is clear that when it is necessary to reduce round off error and increase the speed of polynomial computation the use of Homer's scheme give a stable result. In some cases, when division is faster arried out relatively to multiply operations, usually better to use continued fraction for approximation. For example the IBM-360 use continued fraction FORTRAN-library. This is due to the fact that with the use of bi-linear rational expression in form of continued fraction the general quantity of operation is reduced (Higham, 1997). Major parameters when computing different functions are:

- computational time
- hardware instruction expenditure (memory and processor)
- accuracy of estimation
- bandwidth of argument change
- possibility of calculating with an increase in word length

The choice of one or all these parameters in implementation is called system approach. Here there are two approaches to increase the efficiency: algorithmic and structural organisation.

Quadratic equations and continued fractions: Many problems, when modelled in mathematics, involve a quadratic equation (Knott, 2008) - i.e., an equation of the form:

$$A\ x^2 + B\ x + C = 0 \quad (8)$$

where the A, B and C are numbers and we want to find values for x to make the equation true. For instance, taking $x^2 – 5\ x – 1 = 0$. We can rewrite the equation in a

different way as:

$$x^2 = 5x + 1 \quad (9)$$

and now dividing both sides by x to get:

$$x = 5 + 1/x \quad (10)$$

This means that wherever we have "x", we can replace it by "5 + 1/x". So we can replace the x in "5 + 1/x" for example to get:

$$x = 5 + 1/x = 5 + 1/(5 + 1/x) \quad (11)$$

We can clearly replace the x again and again and get an infinite (periodic) continued fraction:

$$x = 5 + 1/x = 5 + 1/(5 + 1/x) = ... = [5; 5, 5, 5, ...] \quad (12)$$

STRUCTURAL ORGANIZATIONAL PROBLEMS OF APPROXIMATION METHODS

The solution of structural-organisational problems we have found in the use of semi on-line mode of computation. On-line, or semi on-line computation mode is one of the most perspective approach in the present time, where operations are carried out on operand even before the whole word length are formed that is "bit by bit". This principle opposes the traditional off-line mode where an operation can be started only after forming the whole word length (all bits) of the operand. Semi on-line mode allows for the whole formation of word length of one operand and the partial formation of others, such that i-th result is formed before (i+1)-th digit of second operand.

In structural-organizational sense the realization of semi online mode defines a pipeline-like system of collecting computing units working in parallel and forming a general organization in which every i-th unit gets new work not only from the (i-l)-th block but also from (i+k)-th blocks where k = (0, 1,....., m-i+1) and m - number of blocks in the system. Whereas in semi on-line mode *k* is constant in on-line *k* is variable. From this, the build up of a pipelined processor using semi on-line mode requires a feedback from (i+k)-th block to i-th block.

From functional viewpoint, on-line and semi on-line modes defines the change from unary and binary operations to operations of generalized operators, called generalized operators of computing algebraic equations. The major difficulty in implementing them is connected to the fact that part of input operand is not presented wholly (only some digits are present). This results in the presentation of computational results in a redundant number system, which allows the multi-value (ambiguous) representation of number codes (bits). This ambiguity allows certain freedom in representation of every bit of number code got from the result.

With the increase in ambiguity of number system, correspondingly the degree of freedom increases in the choice of next digit. However simultaneously there is an increase in the hardware expenditure, necessary for the implementation of generalized operations. Therefore in this work a quasi-canonical redundant number system where every result digit can be formed from the set {-1, 0, l} is chosen in view of hardware expenditure.

The solution of structural-organisation problem can be found in pipelined organisation of computing process of continued fractions on computers. In this case functions to be computed are expressed in the form of recurrent relations in accordance with the algorithms of Homer's scheme and continued fraction. Recurrent relations (3) and (6) shall be re-written for the purpose of this work in the following forms:

$$F1\left(\left(\ldots\left(Z1\frac{X1}{Y1}+Z2\right)\frac{X2}{Y2}+\ldots+Z_{n-1}\right)\frac{X_{n-1}}{Y_{n-1}}+Z_n\right)\frac{X_n}{Y_n} \quad (13)$$

$$F2 = A_n / (B_n + A_{n-1} / (B_{n-1}+\ldots+A_2 / (B_2 + A_1 / B_1)\ldots)) \quad (14)$$

These recurrent relations shall be defined as pipelined functions where Fl could be seen as a generalized form of Horner's scheme and F2 as a possible form of continued fraction. All the two functions are widely used in computational mathematics. As earlier mentioned the most optimal organization for implementing semi on-line and on-line mode of computing is pipelining. This calls for the design and development of such pipelined structures, which give high efficiency in the solution of not only these problems but also all computational problems of serial characters.

Various works where the use of pipelined devices has been employed based their result on separate implementation of these functions (Zlatev, 1982 and Gustavson, 1972). However these computing devices are mono-functional, and this makes them to be of low efficiency in the process of solving different pipelined functions. The design of multifunctional (combining the functions F1 and F2) processor for the realisation of various pipelined functions is therefore of great interest.

CONCLUSION

From an analysis of the possible methods for implementing continued fraction, the use of Homer's scheme give a stable result when it is necessary to reduce round off error and increase the speed of polynomial computation. In some cases, when division is faster carried out relatively to multiply operations, usually better to use continued fraction for approximation.

ACKNOWLEDGMENT

The efforts of previous researchers in the area of mathematical analysis and especially in numerical analysis are hereby acknowledged. In particular the support of staff at the Department of Mathematics and Statistics of Joseph Ayo Babalola University is hereby acknowledged.

REFERENCES

Al-Kurdi, A.H., R.C. Mittal and D.R. Kincaid, 2002. Performance of various preconditioned GCR(k) algorithms applied to sparse nonsymmetric linear systems, Int. J. Appl. Sci. Comp., 9: 11-31.

Allenby, J.T. and E. Redfern, 1989. Introduction to Number Theory with Computing. Edward Arnold Publishers Ltd., Sevenoaks, Kent.

Golub, G. and C. van-Loan, 1989. Matrix Computation, 2nd Edn., Johns Hopkins University Press, Baltimore.

Gustavson, F.G., 1972. Some Basic Techniques For Solving Sparse Systems Of Linear Equations, In: Rose, D.J. and R.A. Willoughby, (Eds.), Sparse Matrices And Their Applications, Rose, Plenum Press, New York, pp: 41-52.

Higham, N.J., 1997. Iterative refinement for linear systems and lapack, IMA J. Numer. Anal., 17: 495-509.

Knott, R., 2008 An Introduction to the ContinuedFraction. Retrieved from: http://www.mcs.surrey.ac.uk/ Personal /R. Knott/ Fibonacci /cfINTRO.html#refs.

Wheeler, R.F., 2002. Continued Fractions (A Popular Survey). Published by The Mathematical Association.

Wilkinson, J.M., 1948. Progress report on the automatic computing engine, Report MA/17/1024, Mathematics Division, Department of Scientific and Industrial Research, National Physical Laboratory, Tenddington, U.K.

Zlatev, Z., 1982. Use of iterative refinement in the solution of sparsev linear systems, SIAM. J. Numer. Anal., 19: 381-399.