Error Analysis in Numerical Algorithms

Shawki. A. Abbas
Department of Mathematics / College of Science / University of Baghdad

Received in: 9 May 2012 , Accepted in : 20 November 2012

Abstract

In this paper, we applied the concept of the error analysis using the linearization method and new condition numbers constituting optimal bounds in appraisals of the possible errors. Evaluations of finite continued fractions, computations of determinates of tridiagonal systems, of determinates of second order and a "fast" complex multiplication. As in Horner's scheme, present rounding error analysis of product and summation algorithms.
The error estimates are tested by numerical examples. The executed program for calculation is "MATLAB 7" from the website "Mathworks.com"

Key words : Horner's scheme's, tridiagonial systems, linearization method

Primarily

Horner's method can be used to convert between different positional number systems. In which case x is the base of the number system, and the a_j coefficients are the digits of the base x representation of a given number and can also be used if x is a matrix, in which case the gain in the computational efficiency is even greater. In fact when x is a matrix, further acceleration is possible which exploits the structure of matrix multiplication, and only $\sqrt{n}$ instead of n multiplies are needed (at the expense of requiring more storage) using the 1973 method of Paterson and Stockmeyer [1].

For example, to find the product of two numbers , (0.15625)and m.

$(0.15625)\, m = (0.00101_b)\, m = (2^{-3}+2^{-5})\, m = (2^{-3})\, m + (2^{-5})\, m$

$= 2^{-3}\,(m + (2^{-2})\, m) = 2^{-3}\,(\, m + 2^{-2}\,(m)\,).$

Introduction

In a series of textbooks and papers the linearization method is explained and applied to simple examples, in particular, associated error estimates had been missing. These questions have now been thoroughly studied and answered in our perturbation theory for evaluation algorithms of arithmetic expression [2]. The theortical results are applied in this paper to a series of elementary but important algorithms and, at the same time, tested numerically. The results show that our new condition numbers yield much more detailed informations on possible errors than wilkinson's [3] backward analysis.

We analyze the computation of sequences of partial products and sums. The computation of partial products is always a well – conditioned algorithm, while in the class of elementary one – step algorithms is introduced. In particular the associated absolute and relative a priori and a posteriori, condition numbers are established. The error analysis of Horner's scheme shows that our posteriori condition numbers in essence, coincide with error bounds described by Adams [4]. The Taylor & the Newton form of a polynomial. Another important member of the class of elementary one- step algorithms is the Horner like algorithm for evaluating finite continued fractions. A first numerical example deals with rounding errors in the evaluation of partial fractions for $z/\tan(z)$. Next the recursive computation of determinates of tridiagonal matrices is analyzed and the error estimate tested numerically with success by a matrix of l00 rows. In comparison with the stability constants of Babuska [5] for solving tridiagonal linear system.

Our condition numbers constitutes explicit measures for the accumulated errors.

Finally, with the evaluation of determiants of second order. As an application the numerical stability of the common complex multiplication and a "fast" complex multiplication are analyzed and compared.

The detailed rounding error analysis of numerically solving two linear equations in two unknowns has already been established [6]. An error analysis of Gaussian elimination for general linear systems is in preparation. The rounding error analysis of difference and extrapolation schemes is found by our method [7].

The study of numerical algorithms proceeds in the following way first the sequence $f = (f_0, \cdots, f_n)$ of input and arithmetic operations is determined, specifying the algorithm. Then the graph of the functional dependences, the paths of error propagation, and associated weights are described. The system of linear error equations can easily be read from the graph of the algorithm. Its solutions yield approximations of the absolute and relative a priori and a posteriori errors, respectively.

That in this case the condition numbers can easily be determined by recursion formulae.

Let $u=(u_0,\cdots,u_n), \upsilon=(\upsilon_0,\cdots,\upsilon_n)$ denote the sequences of input data, intermediate and final results and their numerical approximations. By $\Delta u_t=\upsilon_t-u_t$ are meant the absolute and by $Pu_t=\Delta u_t/u_t$ the relative a priori errors of the approximations υ_t *of* u_t . The fundamental a priori error estimates then read

$$\left|\Delta u_t\right|\le\sigma_t\eta+O_t(\eta^2)\ ,\ \left|Pu_t\right|\le\rho_t\eta+O_t(\eta^2),\ \text{t=0,…,n} \qquad \ldots(1)$$

where σ_t , ρ_t are the absolute and relative a priori condition numbers and η is the floating point accuracy constant.

The numerical examples in the paper are computed in the decimal floating point arithmetic. The symmetric rounding function rd_N is performed by rounding the 10 digit results to N decimal places. N digits decimal floating point arithmetic is realized by rounding to N places after each operation. The constant η in the numerical examples is thus specified by

$$\eta=5\times10^{-N} \qquad \ldots(2)$$

In general, $\eta<1$ such that the remaindes terms $O_t(\eta^2)$ in (1) can be neglected against the first order terms. In applying the error estimate in (1) to numerical examples, it should be noticed that the condition numbers yield optimal bounds of the possible errors. The actual errors, as a rule, are significantly smaller than indicated by $\sigma_t\eta$, $\rho_t\eta$.

By variation of the parameters, however, we have in most cases, found examples where the actual maximal errors are over estimated at most by factor 5 to 10 such that the magnitude of the error is described correctly.

The sign $\dot{=}$ indicates that the numerical result has been computed in higher precision than the given digits show. A priori condition numbers and a priori errors have always been computed with the highest precision of 10 decimal places.

Computation of Products Algorithm

The methods of our error analysis differ essentially in the number of steps of the idea of algorithm from wilkinson's rounding error analysis (see Wilkinson [3]).

Therefore, the usual product algorithm are discussed thoroughly in this paper. In particular, the condition numbers for perturbations by rounding errors in the arithmetic operations and for data perturbations will be determined and the general error estimates (1) be tested by selected numerical examples.

2.1 Sequences of Partial Products

A finite or infinite product is computed by means of sequence of partial products

$$u_0=b_0\ ,\ u_t=b_tu_{t-1}\quad ,\quad t=1,2,\cdots,n \qquad \ldots(3)$$

Such that

$$U_t=\prod_{j=0}^{t}b_j \qquad \ldots(4)$$

Introducing the indeterminate steps $u_{t-\frac{1}{2}}=b_t$, this algorithm is defined, analogously by the sequence of functions

$$F_0(x)=b_0\ ,\ F_{t-\frac{1}{2}}(x)=b_t\ ,\ F_t(x)=x_{t-\frac{1}{2}}x_{t-1}\quad ,\ for\ t=1,\cdots,n \qquad \ldots(5)$$

The associated linear relative a priori error equation read

$r_0 = e_0^b$, $r_{t-\frac{1}{2}} = e_t^b$, $r_t = r_{t-\frac{1}{2}} + r_{t-1} + e_t^x$, or

$$r_0 = e_0^b \ , \ r_t = r_{t-1} + e_t^b + e_t^x \ , \ t = 1, \cdots, n \qquad \ldots(6)$$

To simplify notation, as here so in the following by $e_t^b, e_t^c, \cdots$ are meant relative errors $Pb_t, Pc_1, \cdots$ and by $e_t^x, e_t^+, e_t', \cdots$ the relative rounding errors of the floating point operations $\times, +, /, \cdots$ the graph of this algorithm is a tree, all non zero weights b_{tk}^{rel} are equal to one (see Fig 1.1).

Obviously, the linear system (6) has the solution

$$r_t = e_0^b + \sum_{j=0}^{t} (e_j^b + e_j^x) \quad , t = 1, \cdots, n \qquad \ldots(7)$$

From this representation one immediately derives the associated relative a priori condition numbers ρ_t^D of data perturbations only ($e_j^x = 0$, $|e_j^b| \le \eta$), and ρ_t^R of perturbations by rounding errors in the arithmetic operations only ($e_j^b = 0$, $|e_j^x| \le \eta$):

$$\rho_t^D = t+1 \ , \ \rho_t^R = t \ , \ \rho_t = \rho_t^D + \rho_t^R = 2t+1 \qquad \ldots(8)$$

Consequently

$$\rho_t^R = \rho_t^D - 1 \quad , t = 1, \cdots, n \ , \qquad \ldots(9)$$

So that the computation of partial products is well- conditioned algorithm.

1. Numerical Examples

Example 3.1

$$U_t = (\exp .01)^t \ , \ t = 1,2, \cdots \qquad \ldots(10)$$

This sequence of powers is established by the above algorithm for $b_0 = 1$, $b_t = b = \exp .01 = 1.01$

We compute the sequence numerically in 2 digits decimal floating points. The basis b rounded to 2 digits, becomes $b' = 1.01$ having the relative error $e_t^b = e^b = 4.93 \times 10^{-5}$. In this case, (7) yields the representation

$$r_t = t \times 4.93 \times 10^{-5} + \sum_{j=1}^{t} e_j^x \qquad \ldots(11)$$

Table 1.1.1 contains the relative errors Pu_t divided by t for t=10,20,…,300. It turns out that Pu_t increases, in essence. Linearly with t as te^b. the relative errors e_t^x seem not to participate in the growth of the error.

Table 1.1.1 shows that the mean values over e_t^x cancel to can be estimated by

$$\left| \frac{1}{t} \sum_{j=1}^{t} e_j^x \right| \overset{\bullet}{=} \left| \frac{1}{t} Pu_t - 4.93 \times 10^{-5} \right| \le 5 \times 10^{-7} \qquad \ldots(12)$$

For t= 120,…,300. thus it is seen that mean value sequence (e_j^x) are by a factor of 1×10^{-2} less than the floating point accuracy constant $\eta = 5 \times 10^5$. That is, randomly distributed rounding errors e_t^x cancle to a considerable extent.

Example 3.2

$$\frac{\sin(\pi x)}{\pi x}=\prod_{k=1}^{\infty}(1-\frac{x^2}{k^2}) \qquad \ldots(13)$$

(see Abramowitz-stequn [7,p.75]). The associated sequence of partial products is computed for $x=2.17$, $\frac{\sin \pi x}{\pi x}=\frac{3}{13\pi}=7.35\times10^{-2}$ in the form

$$\upsilon_0=1\ ,\ \upsilon_t=rd_N(rd_N(b_t)\,\upsilon_{t-1})\ ,\ b_t=1-\frac{x^2}{t^2}\ \ ,t=1,2,\cdots, \qquad \ldots(14)$$

In 6- digit decimal floating point such that $\eta=5\times10^{-6}$. The exact partial products U_t are computed approximately in 10 – digit decimal floating point. Table 1.1.2 lists, blockwise, the index t, the numerical approximation υ_t, the associated relative error $\rho u_t=(\upsilon_t-u_t)/u_t$ and the error sums

$$\sum_{j=1}^{t}e_j^b\quad,\quad\sum_{j=1}^{t}\left|e_j^b\right|\quad,\quad\sum_{j=1}^{t}e_j^x\quad,\quad\sum_{j=1}^{t}\left|e_j^x\right| \qquad \ldots(15)$$

The numerical results show that the absolute error sums of the sequences (e_j^b), (e_j^x) increase, in essence, linearly with t. in the error sums, however, the errors e_j^b , e_j^x again cancel to a large extent. Accordingly, the relative errors Pu_t do not grow systematically but remain bounded in modulus by about $8\eta=4\times10^{-5}$.

In addition, it is easily verified using the listed results that the linear error approximations r_t in (7) approximate the relative error Pu_t very accurately.

2.1 The Summation Algorithm

The algorithm $u_0=c_0\ ,\ u_t=u_{t-1}+c_t\ \ ,t=1,\cdots,n$ …(1) yields the sequence of partial sums $u_t=\sum_{j=0}^{t}c_j$ ...(2) specifying the input of coefficients by intermediate steps $u_{t-\frac{1}{2}}=c_t$, the algorithm is defined by the sequence of functions

$$F_0(x)=c_0\ ,F_{t-\frac{1}{2}}(x)=c_t\ ,F_t(x)=x_{t-1}+x_{t-\frac{1}{2}}\ \ ,\ for\ t=1,\cdots,n \qquad \ldots(3)$$

The linear absolute a priori error equations read

$$s_0=c_0e_0^c\ ,s_t=s_{t-1}+c_te_t^c+u_te_t^+\quad ,\ for\ t=1,\cdots,n \qquad \ldots(4)$$

The associated graph (see Fig 1.2) is a tree, all nonvanishing weights b_{tk}^{abs} are equal to one. The absolute a priori condition number σ_t are determined according to refences, with weights $\alpha_{t-\frac{1}{2}}=|c_t|\gamma_t^c\quad,\quad\alpha_t=|u_t|\gamma_t^T$, recursively by

$$\sigma_0=|c_0|\gamma_0^c\quad,\quad\sigma_t=\sigma_{t-1}+|c_t|\gamma_t^c+|u_t|\gamma_t^T\quad,t=1,\cdots,n \qquad \ldots(5)$$

These condition numbers have the expicit representation

$$\sigma_t=|c_0|\gamma_0^c+\sum_{j=1}^{t}(|c_j|\gamma_j^c+|u_j|\gamma_j^T) \qquad \ldots(6)$$

When the errors in the coefficients e_j have the same order of magnitude as rounding errors in the arithmetic operations, the partial condition numbers σ_t^D , σ_t^R of the summation algorithm under data perturbations only, assuming exact arithmetic operations ($\gamma_j^c = 1$, $\gamma_j^+ = 0$), and under rounding errors in the arithmetic operations, assuming exact data ($\gamma_j^c = 0$, $\gamma_j^+ = 1$), become

$$\sigma_t^D = \sum_{j=0}^{t} |c_j| \quad , \quad \sigma_t^R = \sum_{j=1}^{t} |u_j|, \qquad \ldots(7)$$

The associated relative a priori condition numbers are

$$\rho_t^D = \frac{\sum_{j=0}^{t} |c_j|}{\left|\sum_{j=0}^{t} c_j\right|} \quad , \quad \rho_t^R = \frac{\sum_{j=0}^{t} |u_j|}{\left|\sum_{j=0}^{t} c_j\right|} \qquad \ldots (8)$$

Such that $\rho_t = \frac{\sigma_t}{|u_t|} = \rho_t^D + \rho_t^R$ …(9)

As an example, consider the partial sums of the exponential series

$\exp x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots = \sum_{j=0}^{\infty} c_j$ then $\sum_{j=0}^{\infty} |c_j| = \exp|x|$ and for sufficiently large n one has

$$\sigma_n^D = \exp|x| \quad , \quad \rho_n^D = \frac{\exp|x|}{\exp x} = \begin{cases} 1 & , x \geq 0 \\ \exp 2|x| & , x < 0 \end{cases} \qquad \ldots(10)$$

Hence for negative x the condition numbers increase exponentially with increasing $|x|$. Forsythe [13] discuss the example of computing $\exp(-5.5) = 4.08 \times 10^{-3}$ where the terms c_j of the series are computed exactly in floating point to five decimal places such that $|\rho c_j| \leq \eta = 5 \times 10^{-5}$. The summation is extended over so many terms until the first five digits of the partial sums remain unchanged, that is, $rd_5(\upsilon_n) = rd_5(\upsilon_{n+1})$, thus n=25, $\upsilon_n = 2.64 \times 10^{-3}$ and the associated condition number $\sigma_n^D = \exp 5.5 = 2.45 \times 10^2$. From the error estimate (1)in the introduction we obtain that $|\Delta U_n| \leq \sigma_n^D \eta = 1.22 \times 10^{-2}$ whereas the actual absolute error $|\Delta U_n| = 1.45 \times 10^{-3}$. Analogous conditions are met in computing partial sums of the series for $\sin x$, $\cos x$, $\ln x$ etc and similar alternating series.

Example 3.3 In the same way as above, partial sums of the Bessel functions

$$J_0(x) = \sum_{j=} (-1)^j \frac{(x^2/4)^j}{(j!)^2} \qquad \ldots(11)$$

As simultaneously, the absolute a priori condition numbers

$$\sigma_n^D = \sum_{j=0}^{n} \frac{(x^2/4)^j}{(j!)^2} = I_0(x) \quad \ldots(12)$$ are

computed Table 1.2.1 shows some values of $J_0(x)'$ and the absolute errors $\Delta J_0(x) = J_0(x)' - J_0(x)$

In the neighborhood of the fourth zero 11.79 … . The function I_0 increases from 1.54×10^4 to 1.6×10^4 in this interval. Hence $1.6\times10^4 \times 5\times10^{-5} = .8$ is the approximate a priori bound for the absolute errors $\Delta J_0(x)$. This bound is six times the largest error at 11.82

When the terms c_j of the sum (2) are positive we have

$$\rho_t^D = 1 \quad , \quad \rho_t^R \le t \quad \ldots(13)$$

and thus

$$\rho_t^R \le t\,\rho_t^D \quad , t = 1,\cdots,n \quad \ldots(14)$$

Hence the computation of sums with positive terms is always quasi-well-conditioned. In contrast to the condition numbers ρ_t^D, the condition numbers ρ_t^R are not independent of the ordering of the terms c_0 ,…, c_t . in computing the sums (2) in the floating point arithmetic of a computer, the ordering of the terms should be so chosen that the condition numbers σ_n^R, ρ_n^R and thus the bounds in the error estimates (1) become as small as possible. For a sum with positive terms this is achieved if the terms c_j constitute an increasing sequence. Note that the ordering by increasing absolute values is commonly recommended with out any restrictions. However, if both positive and negative terms c_j occur in the sum, this is in general, no longer true as in the following example shows. In this case the terms have to be ordered with respect to a smallest sum of the absolute values of the partial sums.

Example 3.4

$$S = 1.025\times10^3 - 0.9123\times10^3 - 0.9663\times10^2 - 0.9315\times10^3 \quad \ldots(15)$$ where $c_0 = 1.025\times10^3, \cdots, c_3 = -0.9315\times10^3$ see Wilkinson [3,I.25]. Adding $c_0 = 1025, \cdots, c_3 = -0.9315$ one obtains the partial sums and, by (7), (9), the condition numbers

$$u_1 = 112.7 \quad , u_2 = 16.07 \quad , u_3 = 6.755 \quad , \sigma_3^R = 135.5 \quad , \rho_3^R \doteq 20.06 \quad \ldots(16)$$

In 4- digit decimal floating point the summation is performed without rounding errors and gives the result $\upsilon_3 = u_3 = 6.76$

In converse order $c_0 = -9.315, \cdots, c_3 = 1025$, the terms are arranged with respect to increasing absolute values. Then

$$u_1 = -105.945 \quad , u_2 = -1018.245 \quad , u_3 = 6.755 \quad , \sigma_3^R = 1130 \quad , \rho_3^R \doteq 167.4 \quad \ldots(17)$$

These condition numbers are more than eight times larger than the above condition numbers (16). In 4- digit decimal floating point now the approximate sum $\upsilon_3 = 7.000$ is computed having the relative error $\rho u_3 \doteq 3.6\times10^{-2}$ and the approximate error bound $\rho_3^R \doteq 8.4\times10^{-2}$.

Example 3.5

$$u_n = \sum_{k=0}^{n} \frac{1}{(k+1)^2} \qquad \ldots(18)$$

We compute this partial sum of the infinite series of $x^3/6 = 1.645$ for n=1023 in 6- digit decimal floating point. The exact partial sum rounded to six places in u_n=1.6439. the associated absolute a priori condition numbers

$\sigma_0 = |c_0|\gamma_0^c$, $\sigma_t = \sigma_{t-1} + |c_t|\gamma_t^c + |u_t|\gamma_t^+$ t=1,…,n

Are computed recursively from the equation

$\sigma_0 = |c_0|$, $\sigma_t = \sigma_{t-1} + |u_t| + |c_t|$ t=1,…,n …(19)

The relative a priori condition number is finally determined by $\rho_n = \sigma_n / |u_n|$. The numerical summation in natural ordering yields partial sums which are constant for t≥m=446.

V_m =1.64308 , $\rho_m \dot{=} 445$, $\rho u_m \dot{=} 2.29 \times 10^{-4}$ …(20)

The error bound $\rho_m \eta \dot{=} 2.23 \times 10^{-3}$ overestimates the error ρu_m by factor of about 10. in converse order of increasing terms the algorithm computes

V_n =1.644 , $\rho_n \dot{=} 5.57$ …(21)

This value coincides with the above rounded value of u_n in all decimal places. We observe considerable differences in the magnitude of the condition numbers (20), (21).

In context, let us refer already to the result dealing cascade summation. There we shall compute

V_n =1.643 , $\rho_n \dot{=} 11$ …(22)

This approximation differs by one unit of the last decimal place from the rounded u_n. the last condition number (22) lies between the two condition numbers above such that the second summation procedure is best for this series.

Conclusion

The perturbations of function values p(z), due to perturbations of the argument z, have been described already.

Hence we limit the further study to the case of unperturbed arguments, realized in practice by the input of machine numbers as arguments z. the bound for the absolute a posteriori error $|\Delta \upsilon_i| / \eta$ has been stated by Tsao [12,(5.5)]. Who also proposes the recursive computations of the bound. A similar result is found in Adams [4]. Where it is attributed to kahan. The theortieal results of the paper have been applied to a series of concrete algorithms, and have proved to be very effective means of both a priori and a posteriori error analysis.

References

1. Higham, N. J. (2002), " Accuracy and Stability of Numerical Algorithms" , by the society for industrial and applied mathematics.

2. Stummel, F., (1981), "Perturbation Theory for Evaluation Algorithms of Arithmetic Expression", Math. Comput, 37, P. 435-473.

3. Wilkinson, J. H., (1963), "Rounding error in algebraic processes", Prentice Hall, Englewood Cliffs: Prentice Hall.

4. Adams, D. A., (1967), "A Stopping Criterion for Polynomial Root Finding", Comm. ACM 10, P. 655-658.

5. Babuska, I., (1969), "Numerical Stability in Numerical Analysis", Amstrdam. North Holland. Proc. IFIP Congress, 1968 11-23.
6. Stummel, F., (1982), "Rounding Errors in Numerical Solutions of Two Linear Equations in Two Unknowns", Math. Methods Appl. Sci. 4,P. 549-571.
7. Stumnel, F., (1985), "Forward Error Analysis of Gaussian Elimination Part I : Error and Nesidual Estimates", Numer, Math. 46,P. 365-395.
8. Abramowitz, M. and Stegun, I. A., (1967), "Hand Book of Mathematical Functions", New York, Dover 1968.
9. Wilkinson, J. H., (1968), "A Priori Error Analysis of Algebraic Processes", Proc. Int. Congress. Math. Moscow, P. 629-640.
10. Marcozzi, M. Choi, S. and Chen, C. S., (2001), "On the Use of Boundary Conditions for Variational Formulations Arising in Financial Mathematics", Appl. Mat. Comput, 124, P. 197-214.
11. Kaya, D. and EL-Sayed, S. M., (2004), "A Numerical Solution of the Klein-Gordon Equation and Convergence of the Decomposition Method", Appl. Math. Comput, 156, P. 341-353.
12. Dehghan, M., (2006), "A Computational Study of the One-Dimensional Parabolic Equation Subject to Nonclassical Boundary Specifications", Numer, Methods Partial Differential Equations, 22, P. 220-257.
13. Tsao, N. K., (1974), "Some a Posteriori Error Bounds in Floating Point Computations", J. ACM, 21, P. 6-17.
14. Forsythe, G. E. & Moler, C. B., (1967), "Computer Solution of Linear Algebraic Systems", Englewood cliffs: Prentice Hall.
15. Fox, L., (1964), "Introduction to numerical Linear Algebra", Clarendo Press, Oxford.
16. Schonauer, W., (1987), "Scientific Computing on Vector Computers, North Holland, Amstrdam.
17. Thomas, H. Cormen, Charles, E. Leiserson, Roland L. Rivest and Clifford stein (2009)" introduction to Algorithms " 3rd ed. M. T. Press, pp 41, P. 900,990.

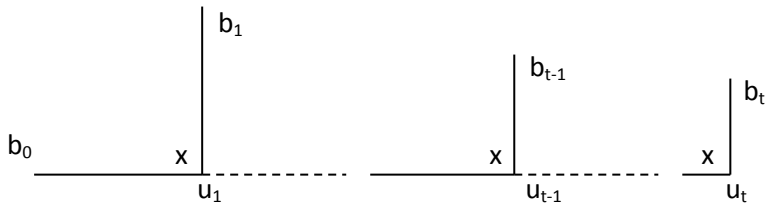

Fig 1: Graph of the linear relative a priori error equations for sequences of partial products

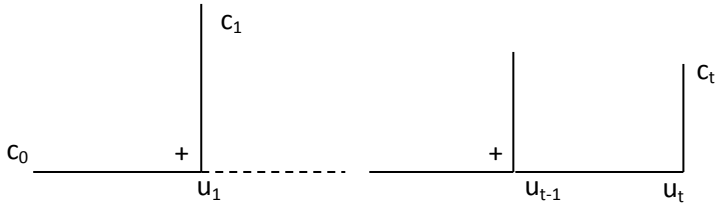

Fig 2: Graph of the linear absolute error equations for the summation algorithm

Table(1.1.1): Relative errors in the computation of $(exp.01)^t$

t	$\frac{10^5}{t}Pu_t$	t	$\frac{10^5}{t}Pu_t$	t	$\frac{10^5}{t}Pu_t$
10	3.9	110	4.9	210	4.9
20	4.5	120	4.9	220	4.9
30	4.6	130	4.9	230	4.9
40	4.7	140	4.9	240	4.9
50	4.8	150	4.9	250	4.9
60	4.8	160	4.9	260	4.9
70	4.8	170	4.9	270	4.9
80	4.8	180	4.9	280	4.9
90	4.8	190	4.9	290	4.9
100	4.9	200	4.9	300	4.9

Table(1.1.2): computation of partial products for $\frac{\sin(\pi x)}{\pi x}$, $\dot{x}=2.17$

100	200	300	400	500
7.69×10^{-2}	7.51×10^{-2}	7.46×10^{-2}	7.43×10^{-2}	7.41×10^{-2}
-6.21×10^{-6}	-1.10×10^{-5}	-1.46×10^{-5}	-2.30×10^{-5}	-3.17×10^{-5}
-3.48×10^{-6}	-1.61×10^{-6}	-4.07×10^{-6}	-3.43×10^{-6}	-2.48×10^{-6}
2.38×10^{-5}	5.05×10^{-5}	7.71×10^{-5}	1.02×10^{-4}	1.27×10^{-4}
-2.72×10^{-6}	-9.43×10^{-6}	-1.05×10^{-5}	-1.96×10^{-5}	-2.93×10^{-5}
5.37×10^{-5}	8.59×10^{-5}	1.20×10^{-4}	1.53×10^{-4}	1.86×10^{-4}

600	700	800	900	1000
7.40×10^{-2}	7.39×10^{-2}	7.39×10^{-2}	7.38×10^{-2}	7.38×10^{-2}
-2.73×10^{-5}	-1.69×10^{-5}	-3.74×10^{-5}	5.87×10^{-6}	-1.49×10^{-5}
-5.35×10^{-7}	-2.54×10^{-6}	-1.36×10^{-6}	8.69×10^{-7}	-1.08×10^{-6}
1.52×10^{-4}	1.77×10^{-4}	2.04×10^{-4}	2.25×10^{-4}	2.49×10^{-4}
-2.69×10^{-5}	-1.44×10^{-5}	-3.61×10^{-5}	5.00×10^{-6}	-1.39×10^{-5}
2.23×10^{-4}	2.56×10^{-4}	2.84×10^{-4}	3.25×10^{-4}	3.71×10^{-4}

Table(1.2.1): Numerical Computation of $J_0(x)$

x	$J_0(x)$	$J_0(x)'$	$\Delta J_0(x)$
11.78	-2.68×10^{-3}	-5.21×10^{-2}	-4.95×10^{-2}
11.79	-3.56×10^{-4}	7.52×10^{-2}	7.52×10^{-2}
11.80	1.97×10^{-3}	-9.17×10^{-2}	-9.37×10^{-2}
11.81	4.29×10^{-3}	-2.47×10^{-2}	-2.89×10^{-2}
11.82	6.61×10^{-3}	1.38×10^{-1}	1.32×10^{-1}
11.83	8.93×10^{-3}	3.72×10^{-2}	2.83×10^{-2}

تحليل الأخطاء في الخوارزميات العددية

شوقي عبد المطلب عباس
قسم الرياضيات / كلية العلوم / جامعة بغداد

أستلم البحث في : 9 أيار 2012 ، قبل البحث في : 20 تشرين الثاني 2012

الخلاصة

في هذا البحث طبقت مفاهيم تحليل الاخطاء الذي استخدمت فيه الطريقة الخطية ، والاعداد الشرطية الجديدة التي تشكل الحد الامثل لتقييم الاخطاء المحتملة. تقدير الكسور المستمرة المحدودة وحساب الانظمة الثلاثية المحدودة للمحددات من الدرجة الثانية، والضرب السريع المركب كما في طريقة هورنر، بوجود تحليل اخطاء التدوير لخوارزميات الضرب والجمع.
تقديرات الاخطاء قد اختبرت بأمثلة عددية ،اما اسم البرنامج الذي استخدم في الحساب فهو " MATLAB 7" .

الكلمات المفتاحية : مخطط هورنر، النظام الثلاثي ، الطريقة الخطية.