Computing 35, 141 – 151 (1985)

Computing

Accurate Arithmetic Results for Decimal Data on Non-Decimal Computers

W. Auzinger and **H. J. Stetter,** Vienna

Received March 18, 1985

Dedicated to Professor R. Albrecht on the occasion of his 60th birthday

Abstract — Zusammenfassung

Accurate Arithmetic Results for Decimal Data on Nondecimal Computers. Recently, techniques have been devised and implemented which permit the computation of smallest enclosing machine number interval for the exact results of a good number of highly composite operations. These exact results refer, however, to the data as they are represented in the computer. This note shows how the conversion of decimal data into non-decimal representations may be joined with the mathematical operation on the data into one high-accuracy algorithm. Such an algorithm is explicitly presented for the solution of systems of linear equations.

Genaue arithmetische Ergebnisse für dezimale Daten auf nicht-dezimalen Computern. In letzter Zeit wurde eine Methodik entwickelt und implementiert, die die Erzeugung kleinstmöglicher Maschinenzahl-Intervalle für die genauen Ergebnisse verschiedenster vielstufiger mathematischer Operationen gestattet. Diese genauen Ergebnisse beziehen sich jedoch auf die Daten so wie sie im Computer dargestellt sind. In der vorliegenden Note wird gezeigt, wie die Konversion dezimaler Daten in nicht-dezimale Darstellungen mit der mathematischen Operation an den Daten in einen einzigen hoch-genauen Algorithmus zusammengefügt werden kann. Für die Lösung von Systemen linearer Gleichungen wird ein solcher Algorithmus im Detail vorgestellt.

1. Introduction

For well-known reasons, accurate results of numerical computations may ordinarily not be expected from digital computers. The development of Numerical Mathematics has largely been motivated by the necessity to obtain numerical results of a *specified* (a priori) or *specifiable* (a posteriori) accuracy in spite of the shortcomings of the digital computer as an instrument for mathematics. For the design of reliable algorithms and their implementation on a given computer, it is highly desirable that round-off can be suppressed over well-defined sequences of operations so that its effect may be clearly discerned from that of other perturbations.

More specifically, let $F(a_1, \ldots, a_n) \in \mathbb{R}$ denote the true result of a specified sequence of arithmetic operations on the data $a_1, \ldots, a_n \in \mathbb{R}$. Then we would like to be able to retrieve the value

$$f := \square F(a_1, \ldots, a_n) \in \mathbb{M}, \tag{1.1}$$

where $\square : \mathbb{R} \to \mathbb{M}$ denotes a given rounding operation into the set $\mathbb{M}$ of machine numbers in which the computation proceeds. (1.1) is often called the *ideal result* in $\mathbb{M}$. (For more details on questions of computer arithmetic, see, e.g., [1].)

Recently, the computation of f of (1.1) for a wide range of composite operations F has become feasible by means of the Subroutine Library ACRITH (cf. [2], [3]); this fact constitutes a major advance for Numerical Mathematics. Normally, the appropriate ACRITH routine will generate two *adjacent* machine numbers f_1 and $f_2 \in \mathbb{M}$ for which it is guaranteed that

$$f_1 \leq F(a_1, \ldots, a_n) \leq f_2 \tag{1.2}$$

holds, which is equivalent to (1.1) for most purposes. The operations F covered by the present release of ACRITH (see [2] or [3]) include the solution of systems of linear algebraic equations and of algebraic eigenvalue problems, polynomial zeros, evaluation of rational expressions, elementary functions, etc.

Naturally, the evaluation and verification of (1.2) refers to the data $a_1, \ldots, a_n$ as they reside in the storage of the computer and are thus available for computation. If one wishes to obtain a guaranteed inclusion of the result of the operation F for sets of data some or all of which are not elements of $\mathbb{M}$, one has to replace data $a_i \notin \mathbb{M}$ by the smallest enclosing machine interval $[\underline{a}_i, \bar{a}_i]$. The bounds f_1 and f_2 of (1.2) are then computed under the assumption that a_i may be any real number from $[\underline{a}_i, \bar{a}_i]$.

If F depends critically on such an a_i – and this is precisely the situation in which we need an arithmetic tool like ACRITH – the best possible bounds f_1 and $f_2 \in \mathbb{M}$ will no longer be adjacent machine numbers, although such bounds would naturally exist for the originally specified values of the a_i. As most digital computers employ a binary (or hexadecimal) machine number set $\mathbb{M}$, trivial decimal numbers like .1 are not in $\mathbb{M}$ and may thus spoil the sophistication of the ACRITH system. This fact greatly diminishes the practical usefulness of the present ACRITH Subroutine Library.

It has been shown in [4] that the dynamic accuracy representation and the algorithm structure which are at the heart of most of ACRITH's problem solving routines permit an automatic coupling of a conversion procedure for decimal data with the evaluation algorithm proper. This will lead to inclusions (1.2) for decimal data $a_1, \ldots, a_n$ (not in the overflow or underflow regime) which are of the same high quality as the ones now generated for machine data. In view of the universal use of the decimal number system in science and engineering, this facility appears as an important supplement to the ACRITH system.

In this paper, we present a first implementation of the ideas of [4]. After a short review of [4], it is explained how an accurate *linear systems solver* for non-machine data may be designed and implemented. As some kind of linear equation solving is at the heart of most ACRITH problem solving routines, this is the key for the extension of these routines to decimal data. Finally, numerical examples are presented and discussed.

2. The Coupling of Data Conversion and High Accuracy Algorithms

In the ACRITH problem solving routines, the following technique permits the generation of tight and guaranteed bounds (1.2) even in ill-conditioned situations:

An initial approximation $f^{(0)}$ to the requested result $F(a_1, \ldots, a_n)$ is recursively improved:

$$f^{(i)} := f^{(i-1)} + \Delta f^{(i-1)}, \quad i = 1, 2, \ldots; \tag{2.1}$$

$\Delta f^{(i)}$ is computed from the defect of $f^{(i)}$ in a suitably chosen problem defining $F(a_1, \ldots, a_n)$. In order that this defect may be obtained with sufficient relative accuracy, the successive corrections $\Delta f^{(i)}$ are not added to the previous approximations but stored separately at first so that

$$f^{(i)} = f^{(0)} + \Delta f^{(0)} + \ldots + \Delta f^{(i-1)} \tag{2.2}$$

appears as

$$(f^{(0)}, \Delta f^{(0)}, \ldots, \Delta f^{(i-1)}). \tag{2.3}$$

With a floating-point representation for the $\Delta f^{(i)}$, this *staggered correction representation* induces a dynamic accuracy for $f^{(i)}$.

Now, an *exact scalar product arithmetic facility* which permits the exact evaluation of an arbitrary scalar product $f \cdot g, f, g \in \mathbb{M}^m$, irrespective of a potential cancellation of leading digits, may be applied. The computation of defects in algebraic problems may be reduced to the computation of a sequence of scalar products, and a scalar product involving $f^{(i)}$ may be expanded into a "longer" scalar product involving the components of (2.3) individually. Thus the virtual accuracy of $f^{(i)}$ present in (2.3) turns into a real accuracy of the defect of $f^{(i)}$, and a further iterative correction of $f^{(i)}$ becomes feasible if necessary.

Finally, the inclusion (1.2) is obtained and verified by a clever use of interval arithmetic in the last correction step. For details see [5], [6].

Stetter's suggestion for the appropriate introduction of decimal data (or rather "non-machine" data) into such an algorithm has been based on the following observation (cf. [4]):

Just as the staggered correction representation of the current approximation $f^{(i)}$ by the components of (2.3) may be introduced into a defect computation through the exact scalar product facility, the representation of some data by a staggered correction format may also be dealt with: The evaluation of a linear defect of the type

$$Af^{(i)} - b, \quad f, b \in \mathbb{R}^n, A \in \mathbb{R}^{n \times n}, \tag{2.4}$$

simply requires the evaluation of longer scalar products when elements of $f^{(i)}$ *and* A are represented by the terms in (2.2) and

$$A = A^{(0)} + \Delta A^{(0)} + \ldots + \Delta A^{(k)}. \tag{2.5}$$

Thus decimal data a may simply be converted into a staggered correction representation

$$a = a^{(0)} + \Delta a^{(0)} + \ldots + \Delta a^{(k)} \tag{2.6}$$

with $a^{(0)}, \Delta a^{(0)}, \ldots, \Delta a^{(k)} \in \mathbb{M}$; naturally, the orders of magnitude of the $\Delta a^{(\kappa)}$ will decrease by factors ulp as κ increases[1]. This introduces a dynamic accuracy also for the data which can thus appear in the defect computation with the necessary level of approximation.

It remains to determine this necessary level of approximation, i.e. the value of k in (2.5). The most practicable approach seems to be the following (cf. [4]):

At first, decimal data $a \notin \mathbb{M}$ are expanded into a basic approximation $a^{(0)}$ and one correction $\Delta a^{(0)}$. From the relative contribution of $\Delta a^{(0)}$ to the preliminary result

$$f = F(\ldots, a^{(0)} + \Delta a^{(0)}, \ldots),$$

the (local) condition of F with respect to a may be estimated. This permits a prediction of the necessary k. Then, a is represented to that level of approximation and the computation of the definitive result proceeds with this representation of the data. (If k has been predicted too low this is now discovered and a further pass must be initiated.) Naturally, in all well-conditioned situations, $k=0$ will be sufficient.

We conclude this section by an illustrative simple example. For better insight, we turn the usual situation around and assume a *decimal*, 5-digit (truncating) *computer* with the use of *hexadecimal data*. Our task is the evaluation

$$F(a) := \frac{a+1}{a^2-2} \quad \text{for} \quad a = 1.6\,\text{AOA}.$$

Since $a = 1.414215\ldots \approx \sqrt{2}$, a five decimal digit representation of a will be utterly insufficient for an accurate determination of $F(a)$.

We consider the following sequence of operations:

$$\begin{aligned} b &:= \text{conv}(a) \\ c &:= b+1 \\ d &:= b^2-2 \\ e &:= c/d = F(a). \end{aligned}$$

[1] ulp = unit of last position = base$^{-\text{mantissa length}}$ in $\mathbb{M}$.

In *Pass 1* we compute basic approximations and first corrections:

$$\begin{aligned} b^{(0)} &= .14142 \times 10^{1} & \Delta b^{(0)} &= .15087 \times 10^{-4} \\ c^{(0)} &= .24142 \times 10^{1} & \Delta c^{(0)} &= \Delta b^{(0)} \\ d^{(0)} &= -.38360 \times 10^{-4} & \Delta d^{(0)} &= .42672 \times 10^{-4}\,(!) \\ e^{(0)} &= .62935 \times 10^{5} \end{aligned} \tag{2.7}$$

$\Delta c^{(0)}$ and $\Delta d^{(0)}$ reflect the influence of $\Delta b^{(0)}$; $\Delta e^{(0)}$ need not be computed because division has a relative condition ≈ 1.

Our *condition number estimates* are:

$$C_{b\to c} = \frac{\Delta c^{(0)}}{c^{(0)}} : \frac{\Delta b^{(0)}}{b^{(0)}} \approx .6$$

$$C_{b\to d} = \frac{\Delta d^{(0)}}{d^{(0)}} : \frac{\Delta b^{(0)}}{b^{(0)}} \approx 1.05 \times 10^{5}\ (!)$$

With $C_{c\to e}$ and $C_{d\to e} \approx 1$ this implies

$$C_{b\to e} = C_{b\to c} \times C_{c\to e} + C_{b\to d} \times C_{d\to e} \approx 1.05 \times 10^{5}.$$

Since the relative round-off error bound in a 5-digit decimal truncating arithmetic is 10^{-4}, this means that we must use $k=1$ in (2.6) in order to obtain a correct 5-digit representation of $F(a)$.

Therefore in *Pass 2*, we employ the following machine representation of $a = 1.6\,\text{AOA}$:

$$a \in b^{(0)} + \Delta b^{(0)} + \Delta b^{(1)} + \overline{\Delta b}$$

where $b^{(0)}$ and $\Delta b^{(0)}$ are as in Pass 1,

$$\begin{aligned} \Delta b^{(1)} &= .89062 \times 10^{-9} \\ \overline{\Delta b} &= [0,1] \times 10^{-14} \qquad \text{(last digit of } \Delta b^{(1)}\text{)}. \end{aligned}$$

This leads to

$$c \in .24142 \times 10^{1} + .15087 \times 10^{-4} + [.89, .90] \times 10^{-9} =: C$$

and permits a sufficiency accurate representation of $d \doteq a^2 - 2$, with the use of the exact scalar product:

$$\begin{aligned} d^{(0)} &= -.38360 \times 10^{-4} \\ \Delta d^{(0)} &= .42672 \times 10^{-4} \\ \Delta d^{(1)} &= .28174 \times 10^{-8} \\ \overline{\Delta d} &= [.26, .56] \times 10^{-13} \end{aligned}$$

(Here $\Delta b^{(0)}$ has been used in the computation of $\Delta d^{(0)}$, $\Delta b^{(1)}$ in that of $\Delta d^{(1)}$ etc.) This representation may be compressed in an obvious way into

$$d \in .43148 \times 10^{-5} + .17441 \times 10^{-10} + [-15., 15.] \times 10^{-15} =: D.$$

Division of the two intervals C and D in their staggered correction representation yields

$$e = c/d \in .55951 \times 10^6 + [.73, .74] \times 10^1$$

or

$$559510 \leq e \leq 559520$$

in the format (1.2).

Note that the exact evaluation of $F(a)$ for a "double precision" interval enclosing the value of a would only have yielded the inclusion

$$559470 \leq e \leq 559850$$

(after rounding to single precision), while the execution of the sequence (2.7) of operations in a 10-digit decimal (truncating) arithmetic ("double precision") generates the poor approximation 559882.9051.

3. Design of a Linear Systems Solver for Decimal Data

To demonstrate the efficiency of our technique in a more realistic situation, we have designed a linear systems solver which computes last bit accurate inclusions of the solution for decimal data on a hexadecimal machine (IBM).

Since we did not wish to spend a good deal of effort on the design of the conversion routine but rather wished to utilize the available ACRITH routines, we assumed the "non-machine" data in the form

$$a = a_1/a_2 \quad \text{or} \quad a_1 * a_2, \tag{3.1}$$

with $a_1, a_2 \in \mathbb{M}$. If a_2 is chosen as a power of 10, this permits the representation of all decimal numbers

$$a = .\alpha_1 \alpha_2 \ldots \alpha_l \times 10^e \tag{3.2}$$

where

$$\begin{aligned} l=7 \quad &\text{and} \quad -2 \leq e \leq 16 \quad \text{for} \quad \mathbb{M} = \mathbb{M}(16,6), \\ l=16 \quad &\text{and} \quad -6 \leq e \leq 38 \quad \text{for} \quad \mathbb{M} = \mathbb{M}(16,14), \end{aligned}$$

i.e. in single and double precision IBM/370 format.

The computation of the staggered correction representations of a from (3.1) proceeds thus (in the case of $a = a_1/a_2$):

$$\begin{aligned} a^{(0)} &:= \square(a_1/a_2) \qquad (\square \text{ means normal rounding into } \mathbb{M}) \\ \Delta a^{(0)} &:= \square(\square\{a_1 - a_2 \cdot a^{(0)}\}/a_2) \\ &\vdots \qquad\qquad \vdots \\ \Delta a^{(k)} &:= \square(\square\{a_1 - a_2 \cdot a^{(0)} - a_2 \cdot \Delta a^{(0)} - \ldots a_2 \cdot \Delta a^{(k-1)}\}/a_2) \\ \Delta A &:= \diamondsuit\{a_1 - a_2 \cdot a^{(0)} - \ldots - a_2 \cdot \Delta a^{(k)}\} \diamondsuit a_2 \end{aligned} \tag{3.3}$$

Here, $\square\{\ldots\}$ denotes the correctly rounded value in $\mathbb{M}$ of the exact scalar product ... and $\diamondsuit\{\ldots\}$ its tightest enclosing interval in $\mathbb{M}$; $\diamondsuit$ means interval division in $\mathbb{M}$. All these operations are readily available in ACRITH.

The algorithm for the solution of $Ax = b$ uses 3 passes:

Pass 1: Preparation

(i) Computation of basic approximations $A^{(0)}$ and $b^{(0)}$.

(ii) Solution of $A^{(0)} x^{(0)} = b^{(0)}$ by a standard elimination procedure.

(iii) Computation of corrections $\Delta A^{(0)}$ and $\Delta b^{(0)}$.

(iv) Computation of a defect $r^{(0)}$ of $x^{(0)}$ for the more accurate data

$$r^{(0)} := \square\{(A^{(0)} + \Delta A^{(0)}) \cdot x^{(0)} - (b^{(0)} + \Delta b^{(0)})\}.$$

(v) Solution of $A^{(0)} \Delta x^{(0)} = -r^{(0)}$ by standard elimination (using the LU-factorization of (ii)).

(vi) Estimation of the relative condition number of the problem *for the given data*:

$$\text{COND} := \max_{|x_j^{(0)}| \neq 0} \left(\frac{|\Delta x_j^{(0)}|}{|x_j^{(0)}|} \right) \times 16^{m+1} \tag{3.4}$$

($m = 6$ or 14 resp. for IBM/370 single or double precision).

(vii) if COND $\begin{cases} < 16^{m-2} & \text{goto (ix)} \quad \text{(well-conditioned)} \\ > 16^{2m} & \text{stop} \quad \text{(extremely ill-conditioned)}. \end{cases}$

(viii) Computation of corrections $\Delta A^{(1)}$ and $\Delta b^{(1)}$.

(ix) Computation of concluding interval corrections, see (3.3).

Pass 2: Iterated defect correction

Here $k = 0$ or 1, depending on whether step (viii) above was skipped or not, $i = 1$ to start with.

(i) Computation of a more accurate defect for the updated approximation to the solution

$$r^{(i)} := \square\{(A^{(0)} + \ldots + \Delta A^{(k)}) \cdot (x^{(0)} + \ldots + \Delta x^{(i-1)}) - (b^{(0)} + \ldots + \Delta b^{(k)})\}.$$

(ii) Solution of $A^{(0)} \Delta x^{(i)} = -r^{(i)}$ by standard elimination.

(iii) Find relative size of latest improvement

$$\text{CHANGE} := \max_{|x_j^{(0)}| \neq 0} \left(\frac{|\Delta x_j^{(i)}|}{|x_j^{(0)}|} \right).$$

(iv) If CHANGE $\geq 16^{-(m+1)}$ do

$$[i := i+1; \text{ if } i < 10 \text{ goto (i) else stop}]$$

For CHANGE $< 16^{-(m+1)}$ we continue to

Pass 3: Establishment of final inclusion of result x

(i) Computation of an inclusion of the remaining defect of $x^{(0)}+\Delta x^{(0)}+\ldots+\Delta x^{(i)}$

$$\overline{r}:=\diamondsuit\{(A^{(0)}+\ldots+\Delta A^{(k)}+\overline{\Delta A})\,.\,(x^{(0)}+\ldots+\Delta x^{(i)})-(b^{(0)}+\ldots+\Delta b^{(k)}+\overline{\Delta b})\}.$$

(ii) Solution of the interval system $(\diamondsuit A)\cdot\overline{\Delta x}=-\overline{r}$ (by means of the appropriate ACRITH routine).

(iii) $$x:=\diamondsuit\{x^{(0)}+\ldots+\Delta x^{(i)}+\overline{\Delta x}\}.$$

Remarks:

1. In the definitions of COND ((vi) of Pass 1) and of CHANGE ((iii) of Pass 2), the restriction $|x_j^{(0)}|\gg 0$ means that the max is to be taken only over components where $|x_j^{(0)}|$ is not too small, say $|x_j^{(0)}|\geq 16^m\cdot x_{\min}$. This is necessary because corrections of values $\approx x_{\min}$ are no longer representable in $\mathbb{M}$. ($x_{\min}$ is the smallest positive number in $\mathbb{M}$.)

2. The condition estimate assumes that the relative sizes of the components of the first data corrections $\Delta A^{(0)}$ and $\Delta b^{(0)}$ are around 16^{-m} (in the average).

3. The possibility of a failure through the "stops" in (vii) of Pass 1 and (iv) of Pass 2 might be further removed by more sophisticated measures. The present version proved to be quite robust as shown by the numerical examples.

4. Numerical Examples

The algorithm of section 3 has been programmed in Fortran with the aid of ACRITH subroutines. The following examples have been run in *single precision* ($m=6$) on an IBM/370. The indicated values of k and i refer to Pass 2.

The results are contrasted with solutions of the same problems which were obtained by the ACRITH interval solver ILSS after replacing the decimal data with their enclosing 1-bit hexadecimal intervals.

Example 1: 5×5-Vandermonde system:

$$A=\begin{pmatrix}1 & 1.1 & \ldots & 1.1^4\\ 1 & 1.2 & & .\\ 1 & 1.3 & & .\\ 1 & 1.4 & & .\\ 1 & 1.5 & \ldots & 1.5^4\end{pmatrix},\quad b=\begin{pmatrix}6.1051\\ 7.4416\\ 9.0431\\ 10.9456\\ 13.1875\end{pmatrix}.$$

Except column 1 and row 5, the data are not in $\mathbb{M}(16,6)$. The true solution is $x=(1,1,1,1,1)^T$. The inclusion

$$\begin{pmatrix}.FFFFFF, & 1.00001\\ .FFFFFF, & 1.00001\\ .FFFFFF, & 1.00001\\ .FFFFFF, & 1.00001\\ .FFFFFF, & 1.00001\end{pmatrix}$$

is obtained with $k=1$ and $i=4$. The condition estimate is COND $\approx .50\times 10^7$.

The result of ILSS, rounded to $\mathbb{M}(10,3)$ is:

$$\begin{pmatrix}.956, & 1.06\\ .842, & 1.14\\ .847, & 1.19\\ .908, & 1.08\\ .985, & 1.02\end{pmatrix}.$$

Example 2: 5×5-Hilbert system (in its original rational form):

$$A=\begin{pmatrix}1 & 1/2 & 1/3 & 1/4 & 1/5\\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6\\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7\\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8\\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9\end{pmatrix}.$$

b was chosen such that the exact solution is $(1,2,3,4,5)^T$. The inclusion

$$\begin{pmatrix}.FFFFFF, & 1.00001\\ 1.FFFFF, & 2.00001\\ 2.FFFFF, & 3.00001\\ 3.FFFFF, & 4.00001\\ 4.FFFFF, & 5.00001\end{pmatrix}$$

is obtained with $k=1$ and $i=4$. The condition estimate is COND $\approx .95\times 10^7$.

The result of ILSS, rounded to $\mathbb{M}(10,3)$, is:

$$\begin{pmatrix}.997, & 1.01\\ 1.94, & 2.05\\ 2.78, & 3.26\\ 3.61, & 4.33\\ 4.84, & 5.19\end{pmatrix}.$$

Example 3: Polynomial evaluation: Compute the value of

$$p(t) := 12192\,t^3 - 32257\,t^2 - 85344\,t + 225799$$

for $t = 2.645752 \notin \mathbb{M}(16,6)$. The exact answer is

$$p(t) = 3.0563905536 \times 10^{-8}.$$

$p(t)$ is the last component of the solution of $Ax = b$, where

$$A = \begin{pmatrix} 1 & & & \\ -t & 1 & & \\ & -t & 1 & \\ & & -t & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 12192 \\ -32257 \\ -85344 \\ 255799 \end{pmatrix}.$$

Our algorithm delivers the accurate 1 bit-inclusion

$$p(t) \in .83455^F_E \times 16^{-6}$$

with $k = 1$ and $i = 3$. The condition estimate is $\mathrm{COND} \approx .27 \times 10^8$.

The result of ILSS, rounded to $\mathbb{M}(10,3)$ is:

$$p(t) \in (-0.814 \times 10^{-1},\ 0.814 \times 10^{-1})!$$

Example 4: Sequence of 5×5-systems of varying condition:

$$A = \begin{pmatrix} 4/3+\delta & -1/3 & \dots & -1/3 \\ -1/3 & 4/3+\delta & & \\ \vdots & \vdots & & \vdots \\ -1/3 & -1/3 & \dots & 4/3+\delta \end{pmatrix}.$$

For $\delta = 0$, A is singular. b is chosen such that the exact solution is $(1/3, 1/3, 1/3, 1/3, 1/3)^T$.

For all $\delta = 1/3 \times 10^{-l}$, $l = 0(1)4$, we obtain the optimal hexadecimal result

$$\begin{pmatrix} .555555, & .555556 \\ .555555, & .555556 \\ .555555, & .555556 \\ .555555, & .555556 \\ .555555, & .555556 \end{pmatrix}.$$

k, i, and the condition estimate COND are given in the following table:

δ	k	i	COND
1/3	0	1	$.35 \times 10^3$
1/30	0	1	$.17 \times 10^4$
1/300	0	2	$.38 \times 10^5$
1/3000	1	2	$.17 \times 10^6$
1/30000	1	3	$.16 \times 10^7$

References

[1] Kulisch, U., Miranker, W. L.: Computer Arithmetic in Theory and Practice. New York: Academic Press 1982.

[2] High-Accuracy Arithmetic Subroutine Library (ACRITH): General Information Manual. IBM Publication No. GC 33-6163-01 (Version 1, Release 2), 1984.

[3] High-Accuracy Arithmetic Subroutine Library (ACRITH): Program Description and User's Guide. IBM Publication No. SC 33-6164-01 (Version 1, Release 2), 1984.

[4] Stetter, H. J.: Sequential defect correction for high-accuracy floating-point algorithms. In: Numerical Analysis (Proceedings, Dundee 1983). Lecture Notes in Mathematics, vol. 1066, 186–202 (1984).

[5] Kaucher, E., Rump, S. M.: E-methods for fixed point equations $f(x) = x$. Computing *28*, 31–42 (1982).

[6] Rump, S. M.: Solving nonlinear systems with least significant bit accuracy. Computing *29*, 183–200 (1982).

Dr. W. Auzinger and
Prof. Dr. H. J. Stetter
Institut für Angewandte und
Numerische Mathematik
Technische Universität Wien
Wiedner Hauptstrasse 6–10
A-1040 Wien
Austria

Printed in Austria

Druck: Paul Gerin, A-1021 Wien