# Higher Order Computer Arithmetic

Siegfried M. Rump

SUMMARY

The floating-point arithmetic on computers is designed to approximate the corresponding operations over the real numbers as close as possible. In this paper it is shown by means of counterexamples that this need not to be true for existing machines. For achieving good numerical results a floating-point arithmetic approximating the real operations as close as possible is probably best. For achieving verifications on computers, at least a precisely defined computer arithmetic is indispensable.

In this paper we first introduce the Kulisch/Miranker theory, which represents a sound basis for computer arithmetic. Each operation is precisely defined and, moreover, is of maximum accuracy. That means, the computed result is the floating-point number of the working precision closest to the infinite precise result. The theory also covers directed roundings allowing computations with intervals. These properties hold true for the floating-point numbers of single and double precision as well as for the vectors, matrices and complex extensions over those.

In the second part of the paper we demonstrate the theoretical basis for what we call 'Higher Order Computer Arithmetic'. This is an inclusion theory allowing the development of algorithms to compute bounds for the solution of various problems in numerical analysis. These bounds are automatically verified to be correct and they are of high accuracy. Very often they are of maximum accuracy, that means the left and right bounds of all components of the solution are adjacent in the floating-point screen. Moreover existence and uniqueness of a solution within the computed bounds is automatically verified by the algorithm. If this verification is not possible, a respective message is given. We develop the theory and give algorithms for the solution of systems of linear and nonlinear equations. As demonstrated by examples even for extremely ill-conditioned problems existence and uniqueness of the solution is verified within bounds of least significant bit accuracy.

## 1. INTRODUCTION

A thorough approach to a standardization of computer arithmetic was given by Kulisch ([Ku69], [Ku71]). A summary of the theory is given in the book of Kulisch : "Grundlagen des numerischen Rechnens" ([Ku76]). In this book Kulisch gives a complete list of the spaces occurring in numerical computation :

| | | | | | | |
|---|---|---|---|---|---|---|
| | | R | $\supset$ | D | $\supset$ | S |
| | | VR | $\supset$ | VD | $\supset$ | VS |
| | | MR | $\supset$ | MD | $\supset$ | MS |
| PR | $\supset$ | IR | $\supset$ | ID | $\supset$ | IS |
| PVR | $\supset$ | IVR | $\supset$ | IVD | $\supset$ | IVS |
| PMR | $\supset$ | IMR | $\supset$ | IMD | $\supset$ | IMS |
| | | C | $\supset$ | CD | $\supset$ | CS |
| | | VC | $\supset$ | VCD | $\supset$ | VCS |
| | | MC | $\supset$ | MCD | $\supset$ | MCS |
| PC | $\supset$ | IC | $\supset$ | ICD | $\supset$ | ICS |
| PVC | $\supset$ | IVC | $\supset$ | IVCD | $\supset$ | IVCS |
| PMC | $\supset$ | IMC | $\supset$ | IMCD | $\supset$ | IMCS |

Figure 1. Spaces in numerical competition

The theory was then extended by the succeeding book of Kulisch/Miranker ([Ku/Mi80]). In the table above R is the set of real numbers, VR are the n-tupels or vectors over the real numbers and MR are the $n^2$-tupels, i.e. n x n - square matrices over the real numbers. Moreover, C are the complex numbers with vectors VC and matrices MC over those. The corresponding interval spaces are denoted by IR, IVR, IMR, IC, IVC, IMC. They are subsets of the corresponding power sets with a canonical imbedding.

For various reasons the operations in these spaces are not exactly executable on computers. Therefore, computing with real numbers has to be approximated by computing in some finite precision, where computation should be simple and rapidly executable. There are a number of such finite precision number systems described in the literature, e.g. logarithmic systems [MaMa73], [SwAl75], rational systems [KoMa83], long integer arithmetic [Fa76]. On a computer usually floating-point systems with mantissa and exponent are used. Research in this area is done in the IEEE group (e.g. 4 articles in COMPUTER, Vol. 14, 1981), described in the Kulisch/Miranker theory [Ku71], [Ku76] and [KuMi81], the work by Matula, Kornerup and others.

We refer to the the set of single precision floating-point numbers on some computer by S. If the accuracy is not sufficient, the user has the possibility to compute in the larger subset D (double precision) of R, where R $\supset$ D $\supset$ S. Over the set S we can define vectors (n-tupels) and matrices ($n^2$-tupels), complex numbers (pairs), vectors and matrices over these pairs as well as the corresponding interval spaces. All these sets are listed in the fourth column of the table above. Finally, the corresponding spaces over the set D are listed in the third column of figure 1.

Possible operations are the inner operations, i.e. operations in one of the listed spaces and the outer operations, i.e. operations between elements of two spaces like multiplication of a real matrix by a scalar or the sum of a real interval vector and complex vector. Counting all possible inner and outer operations yields a surprising number of several hundred.

In the past the definitions of these operations where due to the computer manufacturer. Moreover, not the whole set of operations were predefined but only the four basic instructions addition, subtraction, multiplication and division for single and double precision floating-point numbers. All the other (several hundred) operations had to be designed and implemented by the user. His only tool to do this were the four basic operations. Following Kulisch/Miranker we call this the vertical method according to the following figure.

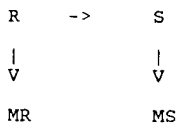



Figure 2. Vertical method

Suppose we want to define the multiplication of two matrices. Then for two n x n - square matrices $A,B \in MS \subset MR$ we have by the mathematical definition

$$(1) \quad (A \times B)_{ij} = \sum_{k=1}^{n} A_{ik} \times B_{kj}$$

for the ij-th component of the product. When implementing this mathematical definition (1) on a computer all additions and multiplications are performed in S. In each operation a rounding error is introduced so that the computed result and the true result may differ significantly. We come to this point later. The matrix multiplication in MS is defined on the computer by the operations in S, following the vertical path in figure 2. However, the definition of the operations in S, i.e. the basic arithmetic operations, is due to the computer manufacturer. And those, this is the experience, do not implement these operations according to mathematical axioms or mathematical theory but more or less by a rule of thumb. For instance one of the leading manufacturers sold several hundred machines doing the following.
The two numbers 134217728.0 and 134217727.0 are exactly representable on that computer. When subtracting both, the result is 2.0 on that computer.

```
  134217718.0
- 134217727.0
-------------
          2.0
```

The relative error of that operation is 1 instead of $10^{-8}$, what it should be. What happens on the computer? The two numbers are in binary representation $1.00 \ldots 0 \times 2^{27}$ and $0.11 \ldots 1 \times 2^{27}$. On the computer the two operands are normalized and the exponent is adjusted. This yields

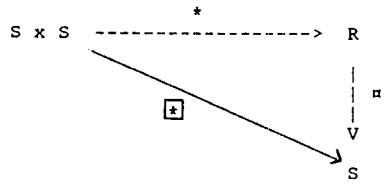
```
              |
0.100 ... 0 | 0 x 2^28
0.011 ... 1 | 1 x 2^28
------------|-----------
0.000 ... 0 | 1 x 2^28
              |
```

However, on the computer the last bit is omitted because of the limited length of the internal accumulator. In other words the accumulator is exactly as long as the mantissa of floating-point numbers. So the result is a 1 one position further to the left, which is a 2 in the binary system.
This is one error found by customers, in fact there are at least $10^8$ such errors on that specific machine. Errors like these are well known (cf. [Fo70], [KaPa77]). There is no guarantee or estimation for the maximum error of a single operation given by the manufactures. Of course, errors like that make a thorough mathematical analysis of algorithms impossible. To obtain an overview on the research work on computer arithmetic the reader is referred to the literature ([Ste74], [Co73], [Ma70]).

We proceed by introducing the definition of computer arithmetic as given in [Ku76] and [KuMi81]. We want to define the four basic operations in S which shall approximate as good as possible the respecting operations in R. The best, of course, would be an isomorphism.

However, R is infinite and S is finite, which makes an isomorphism impossible. But even a mapping with the simpler property of being a homomorphism from R to S is not possible under very weak assumptions. So we go the opposite way and state which properties at least should hold and see, whether they can be satisfied on computers. When applying one of the four basic operations * to two floating-point numbers $a,b \in S$, the result $a * b$ is, in general, an element of R and not of S.

Therefore we first define a mapping $\square : R \to S$, called a rounding. Then we have the operation $* : S \times S \to R$ induced by the real operation, the rounding $\square : R \to S$ and the following diagram:




The result of an operation $\boxed{*} : S \times S \to S$ shall then be defined by the basic property

$$(RG) \quad a,b \in S : \quad a \boxed{*} b := \square(a * b)$$

(cf. [KuMi81]), so that the diagram above commutes. The operations in S are therefore defined by the corresponding operation in R followed by the rounding $\square$. This rounding should satisfy the following properties:

(R1) projection $a \in S$ : $\square a = a$

(R2) monotonicity $a,b \in R$ : $a \leq b \to \square a \leq \square b$

(R4) symmetry $a \in R$ : $\square(-a) = -\square a$

One of the basic results of the Kulisch/Miranker theory is, that the four basic operations $\boxed{*}$ satisfying (RG),(R1),(R2) and (R4) can be realized on computers. Moreover, all operations for the third and fourth column in figure 1 can be effectively implemented on computers satisfying (RG),(R1),(R2) and (R4). In case of interval spaces the operations are inclusion monotone, i.e.

(R3) isotoneness $a \in R$ : $a \subseteq \square a$ ,

where $\square : R \to IS$.

To satisfy especially (RG) in case of vector and matrix spaces an implementation of the precise scalar product is necessary. That means for any two vectors $v,w \in VS$ the scalar product v x w with three roundings (downwards, upwards and to nearest) has to be implemented with maximum accuracy. This is possible using Bohlender's algorithm (cf. [Bo77]).

## 2. LINEAR SYSTEMS

Even with a computer arithmetic with results of maximum accuracy it is possible to produce arbitrary false answers. As a simple example we assume a decimal computer with a two-digit mantissa and consider

(2) $\quad 9 \cdot 14 - 2 \cdot 62 \ .$

Then $9 \cdot 14 = 126$ rounded to two figures results in $13 \cdot 10^1 = 130$, and $2 \cdot 62 = 124$ in $12 \cdot 10^1 = 120$. Therefore on this computer the result is $130 - 120 = 10$ instead of 2. On almost any existing computer the result of

$$10^{50} + 10^{10} - 10^{50}$$

will be zero instead of $10^{10}$. Therefore the associativity of the floating-point operations is not satisfied.

So the implementation of an optimal arithmetic does not suffice to yield only correct results but algorithms deriving error bounds are necessary to achieve true and verifiable results. Of course, an arithmetic satisfying mathematical properties is necessary to approach this goal. In this sense we use the optimally accurate arithmetic as a higher order computer arithmetic. We will continue to demonstrate these algorithms. Let us first give another example.

It is a well known technique first to apply an algorithm to a posed problem and then to slightly altered input data to achieve information about the condition of the problem. Consider

(3) $\quad \begin{aligned} 100\,000\ x + 99\,999\ y &= a \\ 99\,999\ x + 99\,998\ y &= b \ . \end{aligned}$

The following table shows the solution of (3) for different right hand sides (a,b).

| a | b | x | y |
|---|---|---|---|
| 200000 | 200000 | 200000 | -200000 |
| 199990 | 199990 | 199990 | -199990 |
| 200010 | 200010 | 200010 | -200010 |

A small change in the data (the right hand side a and b) causes a small change of exactly the same magnitude in the solution x and y. However, the following table

| a | b | x | y |
|---|---|---|---|
| 199990 | 200010 | 2199970 | -2199990 |
| 200010 | 199990 | -1799970 | 1799990 |

shows a catastrophic effect on the solution x and y for the same magnitude of perturbation in the data. If an algorithm tests automatically the cases of the first table, it would yield the answer that the given problem (3) is very well-conditioned, where, in fact, it is extremely ill-conditioned.

An approach to get rid of that dilemma is to replace every operation by its corresponding interval operation. An interval is a set of the form

$$a,b \in T: \quad [a,b] := \{x \in U \mid a \le x \le b\} \ ,$$

where T is any of the linear spaces S, VS, MS, CS, VCS, MCS or the corresponding spaces over D, and U is the corresponding structure over R. An interval over T is a convex, closed and bounded subset of the power set over U. We use the notation IT for the set of intervals over T. The operations in IT are defined by

$$A,B \in IT: \quad A <*> B := \bigcap \{C \in IT \mid a \in A,\ b \in B:\ a*b \in C\} \ ,$$

where $* \in \{+,-,\cdot,/\}$. We use the symbol <*> according to the Kulisch/Miranker theory. The operations are well defined (with exceptions for /) and effectively implementable for all interval spaces in figure 1.

However, the usage of interval arithmetic may yield an overestimation of the true result. For instance, the result for (2) on a 2-digit computer would be

$$9 <\cdot> 14 <-> 2 <\cdot> 62 =$$

$$[120,130] - [120,130] = [-10,10].$$

This is a true result and displays the ill condition of the problem.

The straightforward replacing of every operation by its corresponding interval operation is called naive interval arithmetic. It turned out that the use of naive interval arithmetic in general yields overestimates of the inclusion of the true result which were sometimes very large. In some cases an algorithm could not finish because a division by an interval containing zero had to be executed. This could happen in cases where ordinary floating-point arithmetic delivered reasonable and accurate approximations.
In the mean time interval mathematics became an independent mathematical discipline, and the times of using only naive interval arithmetic are long ago (cf. [Mo66], [AlHe74], [Al79]).

Next we will demonstrate , how interval arithmetic can be used in another way yielding correct answers with an automatic verification of the correctness and with sharp bounds for the solution. The method will first be developed for the case of linear equations.

Consider the fixed point Theorem of Brouwer:

Theorem 1. Let $f: VR \to VR$ be a continuous function and $0 \neq X \in PVR$ be a closed, bounded and convex subset of VR. If $f(X) \subseteq X$, then there is a fixed point $x \in X$ of f: $f(x) = x$.

Let a linear system

(4) $\quad A\,x = b$

be given for a matrix $A \in MR$ and a vector $b \in VR$. Let Q be an approximate inverse of A, then Qb is an approximate solution of (4). A well known technique to improve this approximation is the residual iteration:

(5) $\quad x_0 = Qb; \quad x_{n+1} = x_n + Q(b - Ax_n) \ .$

Next we will apply a technique to that formula to verify existence and uniqueness of (4) in a certain domain.

Theorem 2. Let the linear system (4) be given, a real matrix Q and a closed, bounded, convex and nonempty subset X of the real numbers. If

(6) $X + Q(b - AX) \underset{\neq}{\subset} X,$

then the matrices A and Q are not singular and the uniquely determined solution of (4) is an element of X.

Remark. The operations in (6) are the power set operations and $X \underset{\neq}{\subseteq} Y$ denotes $X \subseteq Y$ and $X \cap \partial Y = \emptyset$.

Proof. By Theorem 1 there is a fixed point $x$ of $f(x) = x + Q(b - Ax)$ in $X$. Therefore

(7) $x = x + Q(b - Ax)$ and $b - Ax \in \text{Ker } Q$.

Let $y \in \text{Ker } A$. Then for any real number $p$ follows

$$f(x+py) = x + py + Q(b - Ax - pAy) = x + py,$$

i.e. $x + py$ is a fixed point of $f$. If there would be a $0 \neq y \in \text{Ker } A$ this contradicts (6), because there would be a real $q$ with $x + qy \in \partial X$. Therefore $A$ is regular. Suppose $y \in \text{Ker } Q$ and $y \neq 0$. Then $A^{-1}y \neq 0$ and there is a real $q$ with $x + qA^{-1}y \in \partial X$. On the other hand

$$f(x+qA^{-1}y) = x + qA^{-1}y + Q(b-Ax-qy) = x + qA^{-1}y.$$

This contradicts (6) and therefore $Q$ is non-singular. Now (7) implies $b - Ax = 0$ and the theorem is proved.

To apply theorem 2 on computers we use interval operations in IMR and IVR according to the Kulisch/Miranker theory. However, by the rules of interval arithmetic [AlHe74] we obtain

$$d(X + Q(b - AX)) = d(X) + d(Q(b - AX)) \geq d(X) .$$

Therefore (6) can never be satisfied when using interval operations. Another formulation of (6) allows the application of theorem 2 on computers. This is demonstrated by the following theorem.

Theorem 3. With the assumptions of theorem 2 and

(8) $Qb + (I - QA)\cdot X \underset{\neq}{\subset} X$

instead of (6) all assertions of theorem 2 remain valid.

Several significant improvements of that theorem are described in [Ru83] and [Ru84].

For the actual implementation on computers we use the operations in IMS and IVS as defined in the Kulisch/Miranker theory.

In the following, we give a brief description of an algorithm for solving systems of linear equations. The complete algorithm and several further improvements are given in ([Ru83,84]). One essential improvement is the $\varepsilon$-inflation of potential inclusions to accelerate the convergence (cf. step 2 of the succeeding algorithm). It is defined by

$$Y \circ \varepsilon := Y \cdot [1 - \varepsilon, 1 + \varepsilon] .$$

In practice a suitable value for $\varepsilon$ is 0.1 .

1. Use your favorite floating-point algorithm to compute an approximate inverse Q of A.

2. $w = Q\cdot b$; $k := 0$; $Z := Q \boxdot (b - Aw)$; $Y := Z$;
   repeat $k := k+1$; $X := Y \circ \varepsilon$;
   $Y := Z \boxplus (I - Q\cdot A) \boxdot X$
   until $Y \underset{\neq}{\subseteq} X$ or $k > 10$;

3. if $Y \underset{\neq}{\subseteq} X$ then {It has been verified, that there exists one and only one $x \in w \boxplus Y$ with $A x = b$}
   else {No verification}

In step 1 any floating-point arithmetic is applicable, preferably one satisfying (R1), (R2), (R4) and (RG). Of course, the convergence of the algorithm depends on the approximate inverse Q and the spectral radius of I-QA. In step 2 interval arithmetic for higher linear spaces according to the Kulisch/Miranker theory is to be applied.

In practice it turns out, that the algorithm above terminates almost always with $k = 1$. The presented automatic verification is mathematically correct, is not based on plausibility arguments and could replace efforts on the part of the user to make an approximation plausible.

As a numerical example to demonstrate the computational results of the implemented algorithm (cf. [Ru83]) take a linear system with Hilbert matrix of order 21 and right hand side (1,1,....,1). On a computer with 14 hexadecimals in the mantissa (approximately 16 decimals) we achieved inclusions of the solution with maximum accuracy in every component, i.e. the left and right bonds of all including intervals were adjacent in the floating point screen. The new algorithms, moreover, verify the correctness of the result as well as existence and uniqueness of the solution within the computed bounds.

## 3. NONLINEAR SYSTEMS

The ideas of the preceding chapter have been extended to systems of nonlinear equations. Consider a function $f:VR \to VR$ with continuous first derivative. We desire to find small bounds for regions of a zero of $f$. The existence and uniqueness of a zero within those bounds should be automatically verified by the computer. For this purpose consider the following theorem.

Theorem 4. Let $f: VR \to VR$ be a function with continuous first derivative and let $Q \in MR$, $w \in VR$. Denote the Jacobian matrix of $f$ by $f' \in MR$ and for $X \in IVR$ define

(9) $f'(X) := \bigcap \{ Y \in IMR \mid f'(x) \in Y \text{ for all } x \in X \}$ .

If then for some $X \in IVR$

(10) $w - Q\cdot f(w) + \{I - Q\cdot f'(w \boxplus X)\}\cdot(X - w) \underset{\neq}{\subset} X$ ,

then there exists one and only one $x \in X$ with $f(x) = 0$. Moreover the matrix $Q$ and every matrix contained in $I - Q \cdot f'(w \cup X)$ is not singular. $I \in MR$ denotes the identity matrix, $\cup$ denotes the convex union.

The proof of that theorem is given in [Ru83]. According to the proof $f'(w \cup X)$ as defined by (9) cannot be replaced by $\{ f'(x) \mid x \in w \cup X \}$. This can also be demonstrated by counterexamples. Moreover, $f'(w \cup X)$ cannot be replaced by $f'(X)$.

The application of theorem 4 on computers is made possible by evaluating the left hand side of (10) using interval arithmetic (according to the Kulisch/Miranker theory) in higher linear spaces. For example, the multiplication

$$\cdot : MR \times IMR \to IMR$$

is to be used for computing

$$Q \cdot f'(w \cup X)$$

instead of entering a loop

$$S := Q \cdot f'(w \cup X) \text{ with } S_{ij} := \sum_{k=1}^{n} Q_{ik} \cdot f'(w \cup X)_{kj}$$

where $\cdot : R \times IR \to IR$ and $+ : IR \times IR \to IR$ in the summation.

Another important improvement is not to compute an inclusion of a zero itself of the function f but an inclusion of the difference of a zero and an approximation. Consider the following theorem.

Theorem 5. Let $f: VR \to VR$ be a function with continuous first derivative and let $Q \in MR$, $w \in VR$. Let the Jacobian $f' \in MR$ of f be defined by (9). If then for some $X \in IVR$

$$(11) \quad -Q \cdot f'(w) + \{I - Q \cdot f'(0 \cup (w+X))\} \cdot X \subset\!\!\!\!\!\!\neq X ,$$

then there exists one and only one $x \in w + X$ with $f(x) = 0$. Moreover, the matrix $Q$ and every matrix contained in $I - Q \cdot f'(0 \cup (w+X))$ is not singular.

A relative error in the interval vector X plays a less important role in the final inclusion $w + X$. Therefore the final inclusion is sharper than in theorem 4. The computational effort is approximately the same.

The preceding theorems have been extended to the complex number space and to real and complex interval spaces. For example, let a linear system

$$(12) \quad A x = B \quad \text{with} \quad A \in IMR \quad \text{and} \quad B \in IVR$$

be given. The inclusion of the solution of (12) is the set

$$(13) \quad \{ y \in R \mid ay = b \text{ for some } a \in A,\ b \in B \} .$$

The definition applies similarly to nonlinear systems. An inclusion of this set of solutions can be obtained by replacing the respecting matrices, vectors by the respecting interval matrices, interval vectors and the operations by the corresponding interval operations. Therefore, problems where the data is afflicted with errors can be handled. In this case the solution of the problem for any of the infinitely many combinations of input data is included, whereas the computing time is of the same order as a standard floating-point algorithm (the latter, of course, without verification of the result).

Practical experience shows, that the obtained bounds are usually as sharp as possible. In case of point data, almost always maximum accuracy is achieved, i.e. the left and right bounds of all components of the inclusion differ only by one in the last place of the mantissa.

The matrix $Q$ in theorem 4 is an approximate inverse of $f'(w)$, $w$ is an approximation to a zero of $f$. Note, that no further knowledge about the non-singularity of $Q$ or the approximation $w$ is required on the part of the user.

In the following, we give a brief description of an algorithm for solving systems of nonlinear equations.

1. Use your favorite floating-point algorithm
   to compute an approximate zero w of f.

2. Use your favorite floating-point algorithm
   to compute an approximate inverse Q of f'(w).

3. k := 0; Z := $-Q \diamondsuit \hat{f}(w)$; Y := Z;
   repeat k := k+1; X := (Y $\cup$ [0]) $\circ \varepsilon$;
   D := $\hat{f}(w \diamondsuit X)$; Y := Z $\oplus \ominus \{I - Q \cdot D\} \diamondsuit X$
   until $Y \subset\!\!\!\!\!\!\neq X$ or k > 10;

4. if $Y \subset\!\!\!\!\!\!\neq X$ then {It has been verified, that there exists one and only one $x \in w \oplus Y$ with $f(x) = 0$}
   else {No verification}

In steps 1 and 2 any floating-point arithmetic is applicable, preferably one satisfying (R1), (R2), (R4) and (R6). In steps 3 and 4 interval arithmetic for higher linear spaces according to the Kulisch/Miranker theory has to be applied.

Computational results are presented in [Ru83]. The algorithms have been implemented on the UNIVAC 1108 at the University of Karlsruhe, on a Z80 - based minicomputer with 64 kByte memory and on a UNIVAC 1108. It has been applied to nonlinear systems with up to 150 unknowns. The results were almost always of maximum accuracy.

The methods and theorems have been extended to over- and underdetermined linear systems, linear systems with band matrices, sparse linear systems, matrix inversion, algebraic eigenvalue problems, real and complex zeros of polynomials, linear, quadratic and convex programming problems as well as to the evaluation of arithmetic expressions. For all these classes of problems algorithms have been developed and implemented. The achieved results were almost always of maximum accuracy.

There are extensions of these algorithms to complex number spaces, to interval spaces as well as to complex interval spaces.

## 4. CONCLUSION

In the preceding chapters the theoretical background and corresponding algorithms have been given for solving linear and nonlinear systems of equations. Corresponding algorithms for solving linear systems (dense, band, overdetermined, underdetermined and sparse), inversion of matrices, linear programming problems and evaluation of arithmetic expressions compute an inclusion of the solution with automatic verification of correctness, existence and uniqueness; all this is in a self-contained manner. The other algorithms for non-linear systems, algebraic eigenvalue problems, zeros of real and complex polynomials and quadratic and convex programming problems assume an approximation of the solution for use in computing an inclusion of the solution with automatic verification of correctness, existence and uniqueness. This approximation can be obtained by a standard floating-point algorithm. Therefore the latter procedures estimates the error of an approximate solution. In other words they verify the correctness of an error margin (in addition to the verification of existence and uniqueness). These "verification-algorithms" could replace additional tests such as altering input data, recomputing in higher precision etc. Without the new algorithms these tests have to be developed and utilized for each individual problem by the programmer.

The new algorithms perform the verification automatically without any effort on the part of the user, without any knowledge about the condition of the problem and most importantly, without deep mathematical background or an extensive investigation. This, of course, is also true for the algorithms which compute an inclusion of the solution directly without initial approximation. This automatic error control is a key property of all the algorithms presented here. In this sense we use the Kulisch/Miranker arithmetic with maximally accurate single operations as a higher order computer arithmetic, solving basic problems of numerical analysis.

## REFERENCES


[AbBr75] Abbott, J.P., Brent, R.P.: Fast Local Convergence with Single and Multistep Methods for Nonlinear Equations, Austr. Math. Soc. 19 (series B), 173 - 199 (1975).

[Al79] Alefeld, G.: Intervallanalytische Methoden bei nicht-linearen Gleichungen, Jahrbuch Ueberblicke Mathematik 1979, B.I. Verlag, Zuerich.

[AlHe74] Alefeld, G., Herzberger, J.: Einfuehrung in die Intervallrechnung, Bibl. Inst. Mannhein, Wien, Zuerich 1974.

[Bo77] Bohlender, G.: Floating-point computation of functions with maximum accuracy. IEEE Trans. Comput. C-26, No. 7, 621-632, (1977).

[Boe81] Boehm, H.: Automatische Umwandlung eines arithmetischen Ausdrucks in eine zur exakten Auswertung geeignete Form, Bericht des Inst f. Angew. Math., Universitaet Karlsruhe (1981).

[Co73] Cody, J.: Static and Dynamic Numerical Characteristics of Floating-Point Arithmetic, IEEE Transactions on Computers, Vol. C-22, 598-601, (1973).

[CoWe66] Collatz, L., Wetterling, W.: Optimierungsaufgaben, Heidelberger Taschenbuecher, Band 15, Springer Verlag, Berlin, Heidelberg, New York (1966).

[Fa76] Fateman, R.J.: The MACSYMA "Big-Floating-Point" Arithmetic System, Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation, Yorktown Heights, New York, 209-213, (1976).

[Fo70] Forsythe, G.E.: Pitfalls in computation, or why a math book isn't enough, Tech. Rep. No. CS147, Computer Science department, Stanford University, Stanford, California, 1-43.

[KaPa77] Kahan, W., Parlett, B.N.: Can You Count on Your Pocket Calculator, Memorandum No. UCB/ERL M77/21, April 1977, Electronics Research Laboratory, University of California, Berkeley 94720.

[KoMa83] Kornerup, P., Matula, D.W.: Finite Precision Rational Arithmetic, IEEE Transactions on Computers, Vol. C-32, No. 4, April (1983).

[Ku69] Kulisch, U.: Grundzuege der Intervallrechnung, Ueberblicke Mathematik 2, Herausgegeben von D. Laugwitz, graphisches Institut Mannheim, S. 51-98 (1969).

[Ku71] Kulisch, U.: An axiomatic Approach to Rounded Computations, Numerische Mathematik 19, 1-17 (1971).

[Ku76] Kulisch, U.: Grundlagen des numerischen Rechnens, Reihe Informatik, 19, Mannheim/Wien/Zuerich: Bibliographisches Institut (1976).

[KuMi81] Kulisch, U., Miranker, W. L.: Computer Arithmetic in Theory and Practice. Academic Press, New York (1981).

[Ma70] Matula, D.W.: A Formalization of Floating-Point Numeric Base Conversion, IEEE Transactions on Computers, Vol. C-19, No. 8, August (1970).

[Ma75] Matula, D.W.: Fixed-Slash and Floating-Slash Arithmetic, Proc. 3rd Symposium on Computer Arithmetic, IEEE Publ no. 75CH 1017-3C, 90-91 (1975).

[MK80] Matula, D.W., Kornerup, P.: Foundations of Finite Precision Rational Arithmetic, COMPUTING, Suppl. 2, Springer Verlag, 85-111, (1980).

[MaMa73] Marasa, J.D., Matula, D.W.: A Simulative Study of Correlated Error Propagation in Various Finite-Precision Arithmetics, IEEE Transactions on Computers, Vol. C-22, No. 6, June (1973).

[Mo66] Moore, R. E.: Interval Analysis. Prentice Hall (1966).

[Ni66] Nickel, K.: Ueber die Notwendigkeit einer Fehlerschranken-Arithmetik fuer Rechenautomaten, Numerische Mathematik 9, 69-79.

[Ru83] Rump, S.M.: Solving algebraic Problems with high Accuracy, 76 pages, in Proceedings of the "IBM Symposium" on: A New Approach to Scientific Computation, August 1982. Edited by U.W. Kulisch and W.L. Miranker, Academic Press (1983).

[Ru84] Rump, S.M.: Solution of linear and nonlinear algebraic problems with sharp, guaranteed bounds, COMPUTING Supplementum 5, 23 p. (1984).

[Ste74] Sterbenz, H.: Floating-Point Computation, Prentice Hall, 1974.

[SwAl75] Swartzlander, E.E.Jr., Alexopoulos, A.G.: The Sign/Logarithm Number System, IEEE Transactions on Computers, Vol. C-24, No. 12, December (1975).

[St72] Stoer, J.: Einfuehrung in die numerische Mathematik I, Heidelberger Taschenbuecher, Band 105, Springer-Verlag, Berlin, Heidelberg, New York (1972).

[Va62] Varga, R.S.: Matrix Iterative Analysis, Prentice Hall, Englewood Cliffs, New Jersey (1962).