tion φ,

$) \cap Y_{\cdot}^{*} = Y_1^{*}$.

l.

Φ^{-1} are not optimal
s in the accuracy of

equations, connected by
gular points. St. Peters-
in Russian).

the interval integration
pplied problems", Tula,

Interval Computations
No 4(6), 1992

AN APPROACH TO RELIABLE COMPUTATIONS WITH THE MINIMAL REPRESENTATION

Eldar A.Musaev

An approach to organization of computations with guaranteed estimation of the result is proposed. In this schema the process itself choose minimal by complexity representation necessary to provide a priory defined precision.

ПОДХОД К ВЫЧИСЛЕНИЯМ С ГАРАНТИРОВАННОЙ ОЦЕНКОЙ РЕЗУЛЬТАТА С МИНИМАЛЬНЫМ ПРЕДСТАВЛЕНИЕМ

Э.А.Мусаев

Предлагается подход к организации вычислений с гарантированной оценкой результата, при котором вычислительный процесс автоматически устанавливает минимальное по сложности представление, необходимое для получения априорно заданной точности.

1. The problem

Sometimes a continuation of an interval computation comes to a dead end as two values to be compared happens to be incomparable. In the case when indefinity underlies in the precision selected, the idea to repeat computations with a higher one, seems to be very attractive. To get this possibility numerous tools were elaborated to work with different precision and representations (see [1,2,3]).

On the other hand it is impossible to determine a priory what decision is necessary, though it would be highly desirable ([4]). Some approach to get minimal necessary representation automatically is suggested below.

2. Description

The very idea that process of computations could choose the necessary precision was suggested in [5], when the underlyings of concrete way suggested here were described in [6,7,8].

In this model a single program is considered like several programs with similar code and different representations of data. Each such a program we will call later "a shell".

First time the shell with the simplest representation is initialized. If the situation with incomparable values occurs, then this shell is freezed and the control is transfered to the shell with the more complex representation. In terms of concurrent processes it expects the results of more powerful shell.

After a more precize and complex shell computed a critical value more exactly, the frozen low level shall continue work, so in the very end no value is computed more exactly than really needed to get a finite result.

Needless to say that if the second level shell enter to incomparable state, it can wait for a next level shell to solve the problem, and so on.

In this scheme the low level shell make a path for the complex and expensive high level shells, computing, when possible, directions for IF and CASE operators, and a quantity of the iterations for loops. On the other hand the high level shells correct the values computed by the low level shells. That is the reason for the term "wave computations", applied to this schema. Points of control in shells run one after another like waves on a sea.

3. Realization

To shorten explanations below some Algol 68 like language will be used. We draw sample texts for coroutines realization, as concurrent one does not differs seriously.

First, we should fix a set of representations used in computations:

```
_Mode _Level = ( Single, Double, Multi );
_Level FirstLevel = Single, LastLevel = Multi;
```

The data type will contain all necessary representations in this case:

```
_Mode _Num = _Struct ( _SingleSeg s,
                 _DoubleSeg d, _MultiSeg m );
```

In concurrent model appropriate compiler should produce according shell modules.

Global variable GlobLev will be used to point out the current state of the program, and so an addition can be easy expressed like:

```
_Op + = (_Num a,b) _Num: (_Num r;
        ( GlobLev !    s_of r := s_of a + s_of b,
                       d_of r := d_of a + d_of b,
                       m_of r := m_of a + m_of b); r)
```

The comparison operations are not so easy to express because of a possible incomparability in them. In particular they must allow uncertain result, and so should return not usual boolean (True or False) values, but special "Trinary" (True, False and Unknown) values:

```
_Mode _Trin = ( True, False, Unknown );
```

Namely comparison operations realise links between different shells of the computational process:

```
_Op > = (_Num a,b)_Trin:
        _If _Trin f = SeeQueue;
        f <> Unknown           # Already computed ? #
        _Then SetQueue(f,GlobLev); f
        _Elif # This level have not computed yet #
        # Correct all values for a level down ! #
        RecomputeAll;
        # Compute at this level          #
        _Trin f = ( GlobLev ! s_of a > s_of b,
             d_of a > d_of b, m_of a > m_of b );
        f = Unknown           # Not success ? #
        _Then # This level is unsufficient #
```

```
# A level up and compute ... #
JumpUp; JumpDown; SeeQueue
_Else # Success, now save ... #
SetQueue(f,GlobLev); JumpDown; f
_Fi
```

4. Advantages

The main advantage of this scheme is in avoiding extra computations and extra precision in computations. Only necessary precision is used to achieve a result.

On the other hand we may get even better results than computing with higher necessary precision from the very beginning, because lower level shells take some computational work like number of iterations or choice in IF statements. Taking into account that requirements on a time may differ in times for subsequents shells, it could give sufficient economy.

Another advantage of this scheme is that we can place a finite condition on the absolute value of result error, and automatically get it or less one, if that is possible with existing representations.

Besides that, this way can be easy implemented in a real concurrent environment as well as in a coroutines model of computations.

Moreover, besides easiness in getting real concurrent computation, the advantage of this scheme is an easiness in using it with immediate effect.

In development of this scheme a quite exotic representations may be selected, like described in [9] generalized interval arithmetic or a posteriory interval analysis ([10]). In this case representation could influence on a code, but it does not make a matter for the scheme, because the source text of the program may be the same.

References

1. Bohlender, G., Rall, L. B., Ullrich, Ch. and Wolff v. Gudenberg, J. *PASCAL-SC: a computer language for scientific computation.* Academic Press (Perspectives in Computing, vol. 17), Orlando, 1987 (ISBN 0-12-111155-5).

2. *IBM high-accuracy arithmetic subroutine library (ACRITH).* General Information Manual, 1986.

3. Caplat, G. *Syı*
Comp. Sci. **72**

4. Kulisch, U. *Im*
puting **14** (19'

5. Stetter, H. J.
mic precision.
University of S

6. Yakovlev, A.G
tations on con
Siberian Branc
(in Russian).

7. Musaev, E.A.
interval math.

8. Musaev, E.A.
lidation. Inter

9. Musaev, E.A.
"Actual proble
1991, pp. 110–

10. Hansen, E. *A g*
(Lecture notes
pp. 7–18.

11. Matijasevich, '
conference on
1985", vol. 2. (
etc., 1985, pp.

3. Caplat, G. *Symbolic preprocessing in interval function computing.* Lect. Notes in Comp. Sci. **72** (1979), pp. 369–382.

4. Kulisch, U. *Implementation and formalization of floating-point arithmetics.* Computing **14** (1975), pp. 323–348.

5. Stetter, H. J. *Staggered correction representation, a feasible approach to dynamic precision.* In: "Proceedings of the symposium on scientific software", China University of Science and Technology Press, Beijing, 1989.

6. Yakovlev, A.G. *On some possibilities in the organization of localizational computations on computers.* Inf.-operat. material (interval analysis), Computer Center, Siberian Branch of the USSR Academy of Sciences, Krasnoyarsk (1990), pp. 33–38 (in Russian).

7. Musaev, E.A. *Wave computations in interval analysis.* In: "Proc. of seminar on interval math., Saratov, May 29-31, 1990", 1990, pp. 95–100 (in Russian).

8. Musaev, E.A. *Wave computations. A technique for optimal quasi-concurent self-validation.* Interval Computations **1** (3) (1992), pp.53–60.

9. Musaev, E.A. *Non-hierachical wave computations.* In: "Proceedings of conference "Actual problems of modern math." Computer Center of Saratov State University, 1991, pp. 110–112 (in Russian).

10. Hansen, E. *A generalized interval arithmetic.* Nickel, K. (ed.) Interval mathematics. (Lecture notes in computer science, vol. 29), Springer Verlag, New York etc., 1975, pp. 7–18.

11. Matijasevich, Y.V. *A posteriori interval analysis.* In: "EUROCAL '85: European conference on computer algebra, Johannes Kepler Univ., Linz, Austria, Apr. 1–3, 1985", vol. 2. (Lecture notes in computer science, vol. 204), Springer-Verlag, Berlin etc., 1985, pp. 328–334.