Introduction to ACRITH - Accurate Scientific Algorithms

Siegfried M. Rump

0) Abstract. In the past several years new methods have been derived for solving algebraic problems with high accuracy and with automatic verification of the correctness of the result.

The introduction of the new methods consists of three parts:

1. To obtain a result on a computer, the correctness of which is verified to be correct, a precisely defined computer arithmetic is indispensable. Moreover, a computer arithmetic with maximum accurate results for any single operation would be desirable. The definition of maximum accuracy for a computed result is easy, the theoretical and practical foundation is a theory on computer arithmetic developed by U. Kulisch and W. Miranker (cf. [2]) to be described in the following.

   A computer result is an approximation of the precisely defined real result by a floating-point number. Although the real (infinite precise) result is in general not computable, it may serve to define the term maximum accuracy. As long as no overflow or underflow occurs, there are essentially two cases. First, the precise result may be exactly a machine (floating-point) number. Second, there are two floating-point neighbours left and right to the precise real result. In the first case the maximum accurate floating-point result is obviously the exact result which happens to be a machine number. In the second case both floating-point neighbours of the precise result are of maximum accuracy and it depends on the rounding mode which result will be delivered.

   There are four essential rounding modes, namely rounding to nearest, rounding downwards, rounding upwards and rounding towards zero. For rounding to the nearest floating-point number there is the special case that the precise result is exactly the midpoint of the two floating-point neighbours. In this case the result may be defined to be the floating-point number of larger magnitude. The definition of maximum accurate floating-point operations is clear, but the theoretical basis and implementation is not trivial. One reason for this is that the exact, infinite precise result is in general not known.

   However, in numerical analysis not only bare numbers as approximations to real quantities occur but also complex numbers and, more general, vectors and matrices over real and complex numbers. It is desirable to provide the well defined operations for vectors and matrices also with maximum accuracy. Obviously for that purpose a dot product with maximum accuracy is necessary. The implementation of a dot product with maximum accuracy is possible as has been shown by G. Bohlender (cf. [1]). It turns out that the maximum accurate dot product is essentially sufficient to provide all operations for real or complex scalars, vectors and matrices with maximum accuracy (cf. [2]).

   The ultimate goal is the implementation of algorithms delivering results with automatic verification of their correctness. A verification can be provided by delivering sets including the solution. The correctness, i.e. the fact that the delivered set actually contains the solution to the given problem, is to be verified. Therefore a possibility to compute with sets on a computer has to be provided. One way to do this is to use intervals as sets. Therefore, operations have to be defined on the computer for real or complex intervals over scalars, vectors and matrices. Again, all these operations can be performed with maximum accuracy if the dot product is available. This is the reason why the arithmetic of Kulisch and Miranker consists of 5 instead the usual 4 operations, namely +,-,•,/ and the dot product. All the other operations mentioned above can be implemented based on those 5 maximum accuracy.

2. The second part for the introduction of the new methods is a mathematical theory (cf. [3,6]), called inclusion theory. There, theorems are provided the assumptions of which are verifiable on computers with the described arithmetic. The assertions of the theorems state that the given problem is solvable and that the computed inclusion contains this solution. Moreover it is shown that the solution within this set is uniquely determined. Once having verified the

validity of the assumption of one of those theorems, the assertions are true and the result is verified to be correct.

The necessary and sufficient conditions for a set to include the solution of a given problem provided by the inclusion theory are proved by special fixed point theorems. These apply to affine mappings and show the existence of a fixed point of the function in use and the contraction property of this function. Therefore the contraction does not have to be verified a priori by the user but is automatically demonstrated by the theorems and finally by the algorithms based on those. Moreover, the theorems do not suppose the potential inclusion set to be convex *and* compact but the compactness suffices (cf.[6]). Therefore other inclusion sets than rectangular intervals such as torussectors can be used.

The inclusion theory provides theorems for a large number of standard numerical problems such as general systems of linear equations, special linear systems with band or symmetric matrix, matrix inversion, algebraic eigenproblems, polynomial zeros, polynomial evaluation, linear, quadratic and convex programming problems, evaluation of arithmetic expressions, over- and underdetermined linear systems, sparse linear systems, general nonlinear systems of equations and others. The key property of the algorithms based on the inclusion theory is that every result is verified to be correct by means of the automatical proof that the problem is solvable (non-singularity) and by delivering bounds for the solution. In case a problem is not solvable (e.g. inversion of a singular matrix) a respecting message is given.

3. The third part towards the introduction of the new algorithms is the actual implementation. A first implementation has been performed on a UNIVAC 1108. On this computer the arithmetic had to be simulated in software. Then implementations on a Z80 minicomputer and M68000 followed. Since begin of 1984 IBM is offering a Program Product called ACRITH consisting of a subroutine library with the new algorithms, an arithmetic part and an Online Training Component. In ACRITH the arithmetic has been software simulated to run on every S/370 machine. A hardware version is offered for the IBM 4361 processor. The Online Training Component can be used as a training tool to get easy interactive access to the algorithms or as a problem solver to solve problems without having to write a program. Since begin of 1985 a second release of ACRITH is delivered supporting various operating systems (VM, MVS, MVS/XA, VSE/SP, PC/XT370, PC AT/370) and with standard functions for interval input and output.

   In October 1985 a third release of ACRITH has been announced with a large number of new functions such as general systems of nonlinear equations, standard mathematical functions for complex interval arguments with maximum accuracy, complex extensions of other functions, full MVS/XA 31-bit support and others.

The capability to solve even extremely ill-conditioned problems delivering sharp bounds for the solution can be demonstrated by inverting a Hilbert matrix in 21 rows in double precision, that is 14 hexadecimal or 17 decimal figures. The bounds for the elements of the inverse are of maximum accuracy, i.e. they are adjacent floating-point numbers. The result, that the Hilbert matrix is invertible and that the inverse is enclosed within the computed bounds, is automatically verified to be correct.

The computing time required compared with a standard floating-point algorithm is theoretically slower by a factor of 6. This ratio becomes better when using the 4361 processor with the arithmetic in microcode. The computing time for one of the new algorithms using single precision is roughly the time for a traditional algorithm performed in single and in double precision. The latter method is performed frequently to compare the single and double precision results to obtain some indication of the accuracy of the result. Many times a number of control calculations are performed to be more sure of the achieved accuracy. The new algorithms give the automatic verification of the correctness of the computed result.

Following features of the new algorithms are demonstrated by using the Online Training Component of ACRITH, second release. The panels are shown exactly as they appear on the terminal. This introduction gives mathematical and handling information and can be used with or without having access to ACRITH on a computer. 1) Introduction. In science and engineering the use of computers is further expanding. Engineers, physicists, mathematicians and other professionals want to use the computer as a reliable tool. The numbers generated by the computer should be a close approximation to the exact solution of the specified problem.

Users are often completely astonished when presented with the fact that in simple numerical computations with only few operations arbitrarily wrong results may be produced. They are even more astonished to learn that in floating-point computations nothing else can be expected when using standard computation techniques.

To overcome these difficulties a new inclusion theory has been developed (cf. [3,6]). In the IBM Program Product ACRITH algorithms based on the inclusion theory are implemented (cf. [4]). Every result of these algorithms is automatically verified to be correct without any effort on the part of the user.

Computing once in ACRITH in single precision requires approximately the same computing time as running a conventional algorithm in single and double precision. The latter technique is frequently used to get some feeling about the accuracy of the results but giving, of course, no verification of the result. The pure computing times of the two approaches are approximately the same. In addition to that the latter technique requires a lot of human time for programming the double precision version and especially for comparing the single and double precision results.

The program package ACRITH provides a subroutine library for solving a number of standard problems of numerical computation. Moreover an Online Training Component allows to get familiar with the ACRITH functions in an interactive mode.

ACRITH is available under VM/SP, MVS/370, MVS/XA, VSE/SP, PC XT/370, PC AT/370 and is announced under MVS/XA 31-bit mode. The panels shown in this introduction were created under VM/SP. In other operating systems few panels may look a slightly different. For this demonstration package the ISPF Editor was used. Under VM operating system the XEDIT can also be used.

In the following introduction to ACRITH the screens of the ACRITH Online Training Component are displayed exactly as they appear on the terminal. When the user starts ACRITH by typing *ACRITH* and pressing Enter he is provided with panel 1. The next action is typing *s* in the command line. These actions are shown in *italic* above the panels and at the respective position in the panel. Below the panel the next action is shown which is usually to press Enter or to press the PF3 key.

In the following we give a description of the sequence of panels introducing the new techniques and the new algorithms with some mathematical details. Some words or options in the panels are highlighted (like on the terminal). Input data to be entered is displayed in *italic* .

Most of the ACRITH routines in teh subroutine library are provided in single and double precision. In the Online Training Component always the double precision versions are used.

We start with option *s* in panel 1 called to set internal parameters which are shown in panel 16. We choose the ISPF-Editor and switch off the Quick Mode. This will assure, that additional and auxiliary information will be displayed. These are handling informations as well as additional mathematical background. The experienced user being familiar with those may change to the Quick Mode to perform his calculations without being interrupted by help panels.

If PDF is not available, the XEDIT editor can be used as well. Additional information about the editor and the Quick Mode may be obtained by pressing PF1 in panel 16 or 17.

Enter *s* Panel 15

```
----------------- ACRITH Online Training Component (OTC) --------------------
COMMAND ===> s                                              Date     - 85/03/29
                                                            Time     - 23:10
Enter one of the following options and press ENTER .        User     - RUMP
                                                            Terminal - 3279
                                                            PF Keys  - 12
 T  Tutorial          - Information about ACRITH OTC
 S  Set Parameters    - Set parameters for dialog with OTC
 O  ISPF Parameters   - Specify terminal and user parameters
 R  Rounding          - Rounding of decimal and hexadecimal numbers
 D  Dot Product       - Computation of an accurate scalar product
 M  Matrix Mult.      - Multiplication of matrices
 L  Linear Systems    - Solution of linear systems
 I  Inverse           - Inversion of matrices
 E  Eigenproblems     - Computation of eigenvalues and eigenvectors
 P  Polynomial        - Evaluation and zeros of polynomials
 O  Optimization      - Linear programming optimization
 A  Arithm. Expr.     - Evaluation of arithmetic expressions
 V  Vector Operat.    - Vector operations
 F  Std. Functions    - Evaluation of standard functions
 C  Copy              - Copy last input or output data to a file
 X  Exit              - Terminate ACRITH OTC using exit defaults
 ISPF ISPF/PDF        - Invoke ISPF Program Development Facility

Press END KEY (PF3) to terminate                                    PF1=Help
```

Press Enter

Enter *ispf* and *no* Panel 16

```
---------------------------- Set Parameters --------------------------------
COMMAND ===>

   Editor              ===> ispf    ( X Xedit, I Ispf )

   Quick mode          ===> no      ( Y Yes, N No )

Enter HELP command for additional information                       PF1=Help
Press ENTER to process change or END KEY (PF3) to process change and exit.
```

Press Enter

Panel 17

```
---------------------------- Set Parameters --------------------------------
COMMAND ===>

   Editor              ===> ISPF    ( X Xedit, I Ispf )

   Quick mode          ===> NO      ( Y Yes, N No )

Enter HELP command for additional information                       PF1=Help
Press ENTER to process change or END KEY (PF3) to process change and exit.
```

Press PF3

2) **Rounding.** Returning to the main menu the rounding option is called by *r* in panel 18. Rounding is provided from decimal to hexadecimal and vice versa with maximum accuracy.

Three rounding modes are provided: rounding to the nearest floating-point number, rounding downwards and rounding upwards. The rounded downwards result of a decimal number is the largest hexadecimal number in long format being less than or equal to the given decimal number (the ACRITH subroutine library permits rounding to single precision as well).

The hexadecimal long format (same as double precision) consists of a two (hexadecimal) digit exponent and a 14 digit mantissa. The first input is *3.14159* in panel 19. In panel 20 the three rounding modes are displayed in internal format. 41 is the characteristic corresponding to exponent 1 for the hexadecimal mantissa 0.3243F3... .

In the following line the hexadecimal numbers rounded back to 18-digit decimal format are displayed. It can be seen that, when accepting the best (nearest) hexadecimal floating-point approximation for 3.14159, a relative error of $1.2 \cdot 10^{-17}$ is made. Furthermore panel ... shows that the nearest result is closer to the left (rounded downwards) bound, i.e. is smaller than the original decimal input number 3.14159.

Enter *r* Panel 18

```
------------------ ACRITH Online Training Component (OTC) ----------------------
COMMAND ===> r                                              Date     - 85/03/29
                                                            Time     - 23:10
Enter one of the following options and press ENTER .        User     - RUMP
                                                            Terminal - 3279
                                                            PF Keys  - 12
 T  Tutorial          - Information about ACRITH OTC
 S  Set Parameters    - Set parameters for dialog with OTC
 O  ISPF Parameters   - Specify terminal and user parameters
 R  Rounding          - Rounding of decimal and hexadecimal numbers
 D  Dot Product       - Computation of an accurate scalar product
 M  Matrix Mult.      - Multiplication of matrices
 L  Linear Systems    - Solution of linear systems
 I  Inverse           - Inversion of matrices
 E  Eigenproblems     - Computation of eigenvalues and eigenvectors
 P  Polynomial        - Evaluation and zeros of polynomials
 O  Optimization      - Linear programming optimization
 A  Arithm. Expr.     - Evaluation of arithmetic expressions
 V  Vector Operat.    - Vector operations
 F  Std. Functions    - Evaluation of standard functions
 C  Copy              - Copy last input or output data to a file
 X  Exit              - Terminate ACRITH OTC using exit defaults
 ISPF ISPF/PDF        - Invoke ISPF Program Development Facility

Press END KEY (PF3) to terminate                                        PF1=Help
```

Press Enter

Enter *3.14159* Panel 19

```
--------------------------- Precise Rounding -------------------------------
COMMAND ===>

Enter one of the following options in the command line or enter a number in
the 'Input number' field.

   E - Edit input file
   F - Read data from input file


   Input type   ===> DEC  ( D for decimal, H for hexadecimal input)

   Input number ===> 3.14159


A decimal input number is converted to hexadecimal, rounded to the nearest
floating-point number as well as to the smallest enclosing interval
representable in IBM System/370 long floating-point format.

A hexadecimal input number is converted to decimal, rounded to the nearest
floating-point number as well as to the smallest enclosing decimal interval
representable with 18 digits.

Press ENTER to process or END KEY (PF3) to terminate                    PF1=Help
```

Press Enter

Panel 20

```
---------------------- Precise Rounding - Inclusion -------------------------
COMMAND ===>

The decimal input number has been converted to hexadecimal notation, rounded
to the nearest, downwards, and upwards. The result is displayed on the first
output line. Then these hexadecimal numbers have been converted back to
decimal, rounded to the nearest, downwards, and upwards, respectively. The
re-converted decimal numbers are displayed on the second output line.


   Input type was  : DEC

   Input number was: 3.14159


       Nearest                    Downwards                  Upwards
    413243F3E0370CDC           413243F3E0370CDC           413243F3E0370CDD
 0.31415899999999988D+01 ( 0.31415899999999988D+01, 0.31415900000000011D+01)

Additionally the correctness of the displayed results has been algorithmically
verified.

Press ENTER to continue                                                 PF1=Help
```

Press PF3

In panel 21 the editor is called by entering *e* . In panel 22 tutorial information is given how to enter input numbers. This auxiliary information could be turned off by choosing the Quick Mode.

Panel ... shows that the decimal exponent is to be indicated by the letter D.

The following panel 23 shows the input number, a 64 digit decimal number. After entering *f* to read input data from the input file, panel 25 shows the decimal input number rounded to the nearest, downwards and upwards hexadecimal number.

Obviously these three results are identical, i.e. the decimal input number is exactly representable in hexadecimal format. In fact, the input number is exactly 2^{-64}. The example demonstrates that the ACRITH rounding routines provide maximum accurate results even under extreme circumstances.

Enter *e* Panel 21

```
--------------------------- Precise Rounding -------------------------------
COMMAND ===> e

Enter one of the following options in the command line or enter a number in
the 'Input number' field.

   E - Edit input file
   F - Read data from input file


   Input type   ===> DEC  ( D for decimal, H for hexadecimal input)

   Input number ===> 3.14159


A decimal input number is converted to hexadecimal, rounded to the nearest
floating-point number as well as to the smallest enclosing interval
representable in IBM System/370 long floating-point format.

A hexadecimal input number is converted to decimal, rounded to the nearest
floating-point number as well as to the smallest enclosing decimal interval
representable with 18 digits.

Press ENTER to process or END KEY (PF3) to terminate                PF1=Help
```

Press Enter

Panel 22

```
------------------------ Precise Rounding - Edit ----------------------------
COMMAND ===>

Enter one or more decimal numbers to be converted to System/370 floating point
representation (long format) or enter one or more System/370 hexadecimal
floating-point numbers to be converted to decimal.

Example                      0.1234567890
                            -4.34*10**-15
                             98765432.1234567890

A sample input file may look as follows:

EDIT --- ACRITH INPUT A ----------------------------------- COLUMNS 001 072
COMMAND ===>                                               SCROLL ===> HALF
****** ********************************* TOP OF DATA ***********************************
000001 0.1234567890
000002 -4.34D-15
000003 98765432.1234567890
****** ******************************** BOTTOM OF DATA *********************************

Note: Make sure that NUMBER mode is turned off.

Press ENTER to continue or END KEY (PF3) to terminate               PF1=Help
```

Press Enter

Enter *0.000...625* Panel 23

```
EDIT --- ACRITH INPUT A ----------------------------------- COLUMNS 001 072
COMMAND ===>                                               SCROLL ===> PAGE
****** ********************************* TOP OF DATA ***********************************
000001 0.000000000000000000054210108624275221700372640043497085571289062 5
'''''' 
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
****** ******************************** BOTTOM OF DATA *********************************
```

Press PF3

Enter *f*

Panel 24

```
--------------------------- Precise Rounding -------------------------------
COMMAND ===> f

Enter one of the following options in the command line or enter a number in
the 'Input number' field.

   E - Edit input file
   F - Read data from input file


   Input type   ===> DEC  ( D for decimal, H for hexadecimal input)

   Input number ===> 3.14159


A decimal input number is converted to hexadecimal, rounded to the nearest
floating-point number as well as to the smallest enclosing interval
representable in IBM System/370 long floating-point format.

A hexadecimal input number is converted to decimal, rounded to the nearest
floating-point number as well as to the smallest enclosing decimal interval
representable with 18 digits.

Press ENTER to process or END KEY (PF3) to terminate                PF1=Help
```

Press Enter

Panel 25

```
---------------------- Precise Rounding - Inclusion -----  TRUNCATED DISPLAY
COMMAND ===>

The decimal input number has been converted to hexadecimal notation, rounded
to the nearest, downwards, and upwards. The result is displayed on the first
output line. Then these hexadecimal numbers have been converted back to
decimal, rounded to the nearest, downwards, and upwards, respectively. The
re-converted decimal numbers are displayed on the second output line.


   Input type was  : DEC

   Input number was: 0.00000000000000000005421010862427522170037264004349708***


        Nearest                    Downwards                  Upwards
    3110000000000000           3110000000000000           3110000000000000
 0.5421010862427752217D-19 ( 0.5421010862427752217D-19, 0.5421010862427752218D-19)

Additionally the correctness of the displayed results has been algorithmically
verified.

Press ENTER to continue file input                                  PF1=Help
Press END KEY (PF3) to terminate
```

Press PF3

Panel 26

```
--------------------------- Precise Rounding -------------------------------
COMMAND ===>

Enter one of the following options in the command line or enter a number in
the 'Input number' field.

   E - Edit input file
   F - Read data from input file


   Input type   ===> DEC  ( D for decimal, H for hexadecimal input)

   Input number ===> 0.00000000000000000005421010862427522170037264004349708***


A decimal input number is converted to hexadecimal, rounded to the nearest
floating-point number as well as to the smallest enclosing interval
representable in IBM System/370 long floating-point format.

A hexadecimal input number is converted to decimal, rounded to the nearest
floating-point number as well as to the smallest enclosing decimal interval
representable with 18 digits.

Press ENTER to process or END KEY (PF3) to terminate                PF1=Help
```

Press PF3

3) Dot Product. After pressing *PF3* in panel 25 the main rounding panel is displayed and again pressing *PF3* returns to the main menu panel 27. With entering *d* in panel 27 the dot product option in panel 28 is called. Other names for dot product are inner product or scalar product. Entering *e* shows in panel 29 additional information how the two vectors for a dot product have to be entered: first the dimension and then the components of the two vectors. Here interval and point input data (also mixed) are possible. An example is shown in panel 30. It has to be multiplied in vertical direction and to be added in horizontal direction. Multiplication by 1's does not change the first vector in line 2 and adding the components of the first vector yields the exact result 999.

Pressing *PF3* saves the data in the file and pressing Enter in panel 31 shows the results in the following panel. The ACRITH result is exactly 999, in all three rounding modes. The floating-point approximation computed in long format (14 hexadecimal or approximately 17 decimal figures) is instead 768.

The example emphasizes the fact, that the ACRITH dot product delivers results of maximum accuracy under any circumstances. A conventionally calculated dot product may be of poor, in some cases of no accuracy as demonstrated in the example. In contrast the ACRITH dot product maintains the maximum accuracy property even if intermediate overflow or underflow occurs.

Dot products play a key role in scientific computation. On the IBM 4361 processor the arithmetic operations including the dot product are implemented in microcode. Therefore the conventionally calculated dot product (with a loop) and the maximum accurate dot product in ACRITH perform at the same speed. This ability, to calculate dot products with smallest possible error without extra cost opens a wide range of engineering and scientific applications.

Enter *d* Panel 27

```
----------------- ACRITH Online Training Component (OTC) --------------------
COMMAND ===> d                                          Date     - 85/03/29
                                                        Time     - 23:11
Enter one of the following options and press ENTER .    User     - RUMP
                                                        Terminal- 3279
 T  Tutorial          - Information about ACRITH OTC     PF Keys - 12
 S  Set Parameters    - Set parameters for dialog with OTC
 O  ISPF Parameters   - Specify terminal and user parameters
 R  Rounding          - Rounding of decimal and hexadecimal numbers
 D  Dot Product       - Computation of an accurate scalar product
 M  Matrix Mult.      - Multiplication of matrices
 L  Linear Systems    - Solution of linear systems
 I  Inverse           - Inversion of matrices
 E  Eigenproblems     - Computation of eigenvalues and eigenvectors
 P  Polynomial        - Evaluation and zeros of polynomials
 O  Optimization      - Linear programming optimization
 A  Arithm. Expr.     - Evaluation of arithmetic expressions
 V  Vector Operat.    - Vector operations
 F  Std. Functions    - Evaluation of standard functions
 C  Copy              - Copy last input or output data to a file
 X  Exit              - Terminate ACRITH OTC using exit defaults
 ISPF ISPF/PDF        - Invoke ISPF Program Development Facility

Press END KEY (PF3) to terminate                                    PF1=Help
```

Press Enter

Enter *e* Panel 28

```
--------------------------- Precise Dot Product ----------------------------
COMMAND ===> e

For any two vectors the value of their dot product is computed. The results
are algorithmically verified to be correct. For comparison in case of point
input data the value of the dot product computed using long floating point
format is displayed.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random vectors
   B  BROWSE      - Browse input data
   I  INCLUSION   - Obtain inclusion of exact result


Rounding ===>        ( N for nearest, I for interval )

If input data are not exactly representable in long floating-point format
and option I selected, specify whether data should be rounded to the
nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                    PF1=Help
```

Press Enter

Panel 29

```
------------------------ Precise Dot Product - Edit --------------------------
COMMAND ===>

Enter the length of the two vectors and then the two vectors. Their components
may be real numbers or intervals, (-1,0.2) denotes an interval from -1 to 0.2.

Example              Vector A  =  1  2.5  (-1,0.2)
                     Vector B  =  0  -12   3.9


A sample input file may look as follows:


EDIT --- ACRITH INPUT A ------------------------------------ COLUMNS 001 072
COMMAND ===>                                                 SCROLL ===> HALF
****** ********************************* TOP OF DATA ***********************************
000001 3
000002 1 2.5 (-1,0.2)
000003 0 -12  3.9
****** ******************************** BOTTOM OF DATA *********************************

Note: Make sure that NUMBER mode is turned off.

Press ENTER to continue                                             PF1=Help
```

Press Enter

Enter *input* Panel 30

```
EDIT --- ACRITH INPUT A ------------------------------------ COLUMNS 001 072
COMMAND ===>                                                 SCROLL ===> PAGE
****** ********************************* TOP OF DATA ********************************
000001 3
000002 1D19 999 -1D19
000003    1   1     1
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
****** ******************************** BOTTOM OF DATA ********************************
```

Press PF3

Enter *i* Panel 31

```
---------------------------- Precise Dot Product -----------------------------
COMMAND ===> i

For any two vectors the value of their dot product is computed. The results
are algorithmically verified to be correct. For comparison in case of point
input data the value of the dot product computed using long floating point
format is displayed.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random vectors
   B  BROWSE      - Browse input data
   I  INCLUSION   - Obtain inclusion of exact result


Rounding ===>        ( N for nearest, I for interval )

If input data are not exactly representable in long floating-point format
and option I selected, specify whether data should be rounded to the
nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                  PF1=Help
```

Press Enter

Panel 32

```
---------------------- Precise Dot Product - Inclusion ----------------------
COMMAND ===>

The decimal input vectors are exactly representable in hexadecimal format.
For these two vectors the dot product has been computed with three roundings:

         Nearest                     Downwards                    Upwards

     433E700000000000           433E700000000000           433E700000000000
 0.99900000000000000D+03 ( 0.99900000000000000D+03, 0.99900000000000000D+03)

The decimal output numbers on the second line are the converted hexadecimal
numbers rounded to the nearest, downwards, and upwards, respectively.

Additionally the correctness of the displayed results has been algorithmically
verified.

For comparison the result of conventional computation using long format
floating-point operations is displayed:

 0.76800000000000000D+03


Press ENTER to continue                                           PF1=Help
```

Press Enter

Returning to the dot product panel 33 *r* is entered to generate two random vectors for a dot product. In the following panel 34 the length (number of components) *20* is entered and *i* for ill-conditioned. A problem is ill-conditioned if small perturbations of the input data cause large changes of the solution. Here such a problem is randomly generated.

Entering *i* for inclusion in panel 35 gives an error message 'SPECIFY ROUNDING'. Messages are always displayed in the right upper corner of the screen. A rounding has to be specified because the decimal input data is not exactly representable in hexadecimal format. This more detailed information can be obtained by pressing the "help-key" *PF1* . Either rounding to nearest or rounding to the smallest enclosing interval can be specified. In the first case a slightly altered problem is solved, in the second case virtually infinitely many problems are solved, among those the specified problem.

The results can be seen in panel 37. The floating-point approximation, computed in long format, is obviously vastly wrong. Entering *b* in the dot product menu 38 shows the ill-conditioned input data in browse mode in panel 39.

Enter *r* Panel 33

```
---------------------------- Precise Dot Product -----------------------------
COMMAND ===> r

For any two vectors the value of their dot product is computed. The results
are algorithmically verified to be correct. For comparison in case of point
input data the value of the dot product computed using long floating point
format is displayed.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random vectors
   B  BROWSE      - Browse input data
   I  INCLUSION   - Obtain inclusion of exact result


Rounding ===>        ( N for nearest, I for interval )

If input data are not exactly representable in long floating-point format
and option I selected, specify whether data should be rounded to the
nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                  PF1=Help
```

Press Enter

Enter *20* and *i* Panel 34

```
------------------------ Precise Dot Product - Random -------------------------
COMMAND ===>

Random decimal vectors will be generated. They can be displayed in
browse mode. Please specify the length of the two vectors.




   Length     ===> 20      (Length of the two vectors)


To generate an ill-conditioned problem specify I in the "condition" field.

   Condition ===> i        (Specify I for ill-conditioned,
                            or leave the field blank for well-conditioned)




Press ENTER to generate random vectors
Press END KEY (PF3) to cancel request                             PF1=Help
```

Press Enter

Enter *i* Panel 35

```
---------------------------- Precise Dot Product ------ RANDOM DATA GENERATED
COMMAND ===> i

For any two vectors the value of their dot product is computed. The results
are algorithmically verified to be correct. For comparison in case of point
input data the value of the dot product computed using long floating point
format is displayed.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random vectors
   B  BROWSE      - Browse input data
   I  INCLUSION   - Obtain inclusion of exact result


Rounding ===>        ( N for nearest, I for interval )

If input data are not exactly representable in long floating-point format
and option I selected, specify whether data should be rounded to the
nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                  PF1=Help
```

Press Enter

Enter *n* Panel 36

```
---------------------------- Precise Dot Product ----------  SPECIFY ROUNDING
COMMAND ===> I

For any two vectors the value of their dot product is computed. The results
are algorithmically verified to be correct. For comparison in case of point
input data the value of the dot product computed using long floating point
format is displayed.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random vectors
   B  BROWSE      - Browse input data
   I  INCLUSION   - Obtain inclusion of exact result


Rounding ===> n      ( N for nearest, I for interval )

If input data are not exactly representable in long floating-point format
and option I selected, specify whether data should be rounded to the
nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                  PF1=Help
```

Press Enter

Panel 37

```
---------- Precise Dot Product - Inclusion ----------
COMMAND ===>

The decimal input vectors are not exactly representable in hexadecimal format.
They have been converted to the nearest hexadecimal vectors. For these two
vectors the dot product has been computed with three roundings:

        Nearest                    Downwards                  Upwards

     B4B0B6815A8AD132           B4B0B6815A8AD133           B4B0B6815A8AD132
-0.24523842856523090 3D-14 (-0.24523842856523090 8D-14,-0.24523842856523090 2D-14)

The decimal output numbers on the second line are the converted hexadecimal
numbers rounded to the nearest, downwards, and upwards, respectively.

Additionally the correctness of the displayed results has been algorithmically
verified.

For comparison the result of conventional computation using long format
floating-point operations is displayed:

-0.155431223447521916D-14

Press ENTER to continue                                          PF1=Help
```

Press Enter

Enter *b*

Panel 38

```
---------- Precise Dot Product ----------
COMMAND ===> b

For any two vectors the value of their dot product is computed. The results
are algorithmically verified to be correct. For comparison in case of point
input data the value of the dot product computed using long floating point
format is displayed.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random vectors
   B  BROWSE      - Browse input data
   I  INCLUSION   - Obtain inclusion of exact result


Rounding ===> N     ( N for nearest, I for interval )

If input data are not exactly representable in long floating-point format
and option I selected, specify whether data should be rounded to the
nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                 PF1=Help
```

Press Enter

Panel 39

```
BROWSE - ACRITH INPUT A1 ---------- LINE 000000 COL 001 080
COMMAND ===>                                  SCROLL ===> PAGE
********************************* TOP OF DATA ********************************
  20
 -0.958924274663138496    0.940828776112831555    0.241730777970665944
  0.083972219920468286   -0.748098473825360030    0.699905036942435055
 -0.599622615372087206    0.386191062443257571    0.878944023130276531
 -0.500862191772082674    0.111321738796071198   -0.639934433483369688
  0.483770388565062867    0.967178719180435298    0.553192953852895464
  0.667463821418911196   -0.316140497589863081   -0.482224753515135399
  0.054524194225615288   -0.843085508732009700
  0.835855847237526303   -0.858549162480215122    0.853523378122716711
 -0.735799386289737220    0.678692710614756278   -0.418430866013244140
 -0.148251959238790920   -0.920546254730289296    0.912798909766951522
 -0.109720923728396605   -0.973215827205672387    0.949695763469209323
  0.350631140063241767    0.728843966998711984   -0.806753882788402779
  0.789546874046928693   -0.999865245821645626    0.964077328938316575
  0.518814962988764505    0.854506551396388578
******************************** BOTTOM OF DATA ******************************
```

Press PF3

Panel 40

```
---------- Precise Dot Product ----------
COMMAND ===>

For any two vectors the value of their dot product is computed. The results
are algorithmically verified to be correct. For comparison in case of point
input data the value of the dot product computed using long floating point
format is displayed.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random vectors
   B  BROWSE      - Browse input data
   I  INCLUSION   - Obtain inclusion of exact result


Rounding ===> N     ( N for nearest, I for interval )

If input data are not exactly representable in long floating-point format
and option I selected, specify whether data should be rounded to the
nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                 PF1=Help
```

Press PF3

4) Linear Systems. Entering *1* in the main menu 41 enters the linear system option. In panels 42 and 43 a linear system with 10 unknowns is randomly generated. For linear systems interval and point input data (also mixed) are possible.

In panel 44 an inclusion is computed with rounding option *i* . In this case the input numbers are rounded to the smallest enclosing intervals with hexadecimal bounds. Computing the solution of such an 'interval linear system' is equivalent to computing the solution of every linear system contained in the tolerances, i.e. the solution of infinitely many linear systems. With the methods of ACRITH based on the inclusion theory this can be done in a computing time of the same order as a comparable floating-point algorithm would need to solve a single linear system of the same size, the latter, of course, without verification of the correctness of the result. The new algorithms offer the possibility of computing sharp bounds for linear systems with tolerances in the data (cf. [3]).

In panel 45 the solution is displayed. It is shown that e.g. the first component of the solution has the correct digits *1.4929749075382* , where the last two (14th and 15th) digits are between *07* and *12* . This is an inclusion of the solution of every linear system enclosed within the tolerances. The correctness of the bounds has been automatically verified. Additionally the non-singularity of every matrix within the input tolerances has been verified.

Enter *1* Panel 41

```
------------------ ACRITH Online Training Component (OTC) ---------------------
COMMAND ===> 1                                              Date     - 85/03/29
                                                            Time     - 23:13
Enter one of the following options and press ENTER .        User     - RUMP
                                                            Terminal - 3279
 T  Tutorial          - Information about ACRITH OTC         PF Keys  - 12
 S  Set Parameters    - Set parameters for dialog with OTC
 O  ISPF Parameters   - Specify terminal and user parameters
 R  Rounding          - Rounding of decimal and hexadecimal numbers
 D  Dot Product       - Computation of an accurate scalar product
 M  Matrix Mult.      - Multiplication of matrices
 L  Linear Systems    - Solution of linear systems
 I  Inverse           - Inversion of matrices
 E  Eigenproblems     - Computation of eigenvalues and eigenvectors
 P  Polynomial        - Evaluation and zeros of polynomials
 O  Optimization      - Linear programming optimization
 A  Arithm. Expr.     - Evaluation of arithmetic expressions
 V  Vector Operat.    - Vector operations
 F  Std. Functions    - Evaluation of standard functions
 C  Copy              - Copy last input or output data to a file
 X  Exit              - Terminate ACRITH OTC using exit defaults
 ISPF ISPF/PDF        - Invoke ISPF Program Development Facility

Press END KEY (PF3) to terminate                                      PF1=Help
```

Press Enter

Enter *r* Panel 42

```
---------------------- Precise Linear System Solving -------------------------
COMMAND ===> r

The solution of any square linear system is computed. The components of the
solution are displayed rounded to the smallest enclosing intervals.
The results are algorithmically verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random linear system
   H  HILBERT     - Generate linear system with Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise linear system solving


Rounding ===>        ( N for nearest, I for interval )

If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                      PF1=Help
```

Press Enter

Enter *10* Panel 43

```
---------------------- Precise Linear System Solving - Random ----------------
COMMAND ===>

A random decimal linear system will be generated. It can be displayed
in browse mode, altered in edit mode.



Please specify the number of unknowns.

   Unknowns ===> 10     (Number of unknowns of the linear system)


Specify the relative diameter of the components of the linear system
to be generated. The relative diameter of an interval is the positive
difference of its components divided by the maximum of the absolute values
of its two components. Specify 0 for decimal non-interval linear system.

   Diameter ===> 0



Press ENTER to generate random linear system
Press END KEY (PF3) to cancel request                                 PF1=Help
```

Press Enter

Enter *i* and *i* Panel 44

```
---------- Precise Linear System Solving     RANDOM DATA GENERATED
COMMAND ===> i

The solution of any square linear system is computed. The components of the
solution are displayed rounded to the smallest enclosing intervals.
The results are algorithmically verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random linear system
   H  HILBERT     - Generate linear system with Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise linear system solving


Rounding ===> i      ( N for nearest, I for interval )

If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                PF1=Help
```

Press Enter

Panel 45

```
---------- Precise Linear System - Inclusion ----------
COMMAND ===>

The decimal linear system has been converted to the smallest enclosing
hexadecimal interval linear system. Because this system consists of intervals
least significant bit accuracy may not be obtainable. The solution is:


  Component  1   (  0.1492974907538207D+01 ,  0.1492974907538212D+01 )
             2   (  0.1595838107217992D+01 ,  0.1595838107217994D+01 )
             3   (  -0.434135805615590D+00 ,  -0.434135805615586D+00 )
             4   (  -0.310668488854303D+00 ,  -0.310668488854300D+00 )
             5   (   0.278156185498882D+00 ,   0.278156185498887D+00 )
             6   (  -0.999094256268251D+00 ,  -0.999094256268248D+00 )
             7   (  0.1608993080852264D+01 ,  0.1608993080852267D+01 )
             8   (  0.3104594378108572D+01 ,  0.3104594378108577D+01 )
             9   ( -0.3715891757047686D+01 , -0.3715891757047682D+01 )
            10   ( -0.4325439996283333D+01 , -0.4325439996283329D+01 )

Additionally the non-singularity of every matrix contained in the
interval matrix and the correctness of the displayed results have been
algorithmically verified.

Press ENTER to continue                                         PF1=Help
```

Press Enter

Enter *h* Panel 46

```
---------- Precise Linear System Solving ----------
COMMAND ===> h

The solution of any square linear system is computed. The components of the
solution are displayed rounded to the smallest enclosing intervals.
The results are algorithmically verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random linear system
   H  HILBERT     - Generate linear system with Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise linear system solving


Rounding ===> I      ( N for nearest, I for interval )

If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                PF1=Help
```

Press Enter

In the panel 46 option *h* is called to generate a linear system with a Hilbert matrix. Hilbert matrices are extremely ill-conditioned. In panel 47 *PF1* is pressed to obtain tutorial information about Hilbert matrices, displayed in panels 48 through 51. In panel 52 a linear system with Hilbert matrix in 21 unknowns with random right hand side is generated, the condition number of which is $5 \cdot 10^{23}$.

The inclusion of the solution is displayed in panel 54. Pressing Enter shows the full set of inclusions in browse mode. The inclusion is of maximum accuracy, i.e. left and right bounds are adjacent floating-point numbers. In some cases the difference is two decimal units. Despite the fact that this would still be a very sharp inclusion the internal (hexadecimal) inclusion is indeed of maximum accuracy. Only the rounded results in 16 decimal digits are of diameter 2.

Panel 47

```
---------------------- Precise Linear System - Hilbert -----------------------
COMMAND ===>

A linear system with a Hilbert matrix will be generated. The right hand side
will be randomly generated. The system can be displayed in browse mode,
altered in edit mode.


Specify the number of unknowns of the Hilbert linear system, 21 is maximum.

   Unknowns ===>        (Number of unknowns of the Hilbert linear system)


Specify the relative diameter of the components of the system to be generated.
The relative diameter of an interval is the positive difference of its two
components divided by the maximum of the absolute values of its two components.
Specify 0 for a linear system with exact Hilbert matrix.

   Diameter ===> 0


Press ENTER to generate Hilbert linear system
Press END KEY (PF3) to cancel request                              PF1=Help
```

Press PF1

Panel 48

```
TUTORIAL ---------------- OPTION 'L' - LINEAR SYSTEMS ---------------- TUTORIAL
COMMAND ===>                                                  (Page 1 of 4)



The solution of a square linear system of equations is computed.


The linear system can be entered with the editor. This is done by choosing
option E . An intermediate help panel may be displayed depending on the
setting of the quick mode (see option 'S' Set parameters). It is also possible
to generate random input data by selecting option 'R' , or to generate a
Hilbert Matrix by selecting option 'H'.









Press ENTER to proceed to the next page
PF Keys: 1=Help    3=End 4=Return       7=Up 8=Skip      10=Back 11=Next 12=Cursor
```

Press Enter

Panel 49

```
TUTORIAL ---------------- OPTION 'L' - LINEAR SYSTEMS ---------------- TUTORIAL
COMMAND ===>                                                  (Page 2 of 4)

A Hilbert matrix is a special matrix of the form:

    1/1       1/2       1/3       . . .     1/n

    1/2       1/3       1/4       . . .     1/(n+1)

    1/3       1/4       1/5       . . .     1/(n+2)

     .         .         .                   .
     .         .         .                   .
     .         .         .                   .

    1/n       1/(n+1)   1/(n+2)   . . .     1/(2*n-1)

Since this matrix contains numbers which are not exactly representable in
System/370 long floating-point format, the matrix generated by option  H
is the Hilbert matrix multiplied by the least common multiple of all
denominators.

Press ENTER to proceed to the next page
PF Keys: 1=Help    3=End 4=Return       7=Up 8=Skip      10=Back 11=Next 12=Cursor
```

Press Enter

Panel 50

```
TUTORIAL ---------------- OPTION 'L' - LINEAR SYSTEMS ---------------- TUTORIAL
COMMAND ===>                                                  (Page 3 of 4)



The Hilbert Matrix has the property of being extremely ill-conditioned.
Ill-conditioned means that it is very difficult to obtain the solution
of a problem of this type. When a Hilbert Matrix is generated as the
left hand side of the linear system the right hand side is randomly
generated.


After input data has been generated, either by using the edit function or by
generating it as a Hilbert Matrix or randomly, the data can then be displayed
using the browse function, option 'B' .









Press ENTER to proceed to the next page
PF Keys: 1=Help    3=End 4=Return       7=Up 8=Skip      10=Back 11=Next 12=Cursor
```

Press Enter

Panel 51

```
TUTORIAL --------------- OPTION 'L' - LINEAR SYSTEMS --------------- TUTORIAL
COMMAND ===>                                                  (Page 4 of 4)


Now that the data is available the ACRITH linear system can be solved by
selecting option I . If some of the input data are not exactly representable
in hexadecimal format you will be required to specify whether the input data
should be rounded to the nearest representable floating point number or to an
interval that includes the exact number. After the selection has been made
pressing enter will cause the ACRITH OTC to generate an inclusion of the
exact result.

The results are usually of maximum accuracy and are always verified to exist
within the given bounds of the result.









PF Keys: 1=Help    3=End 4=Return       7=Up 8=Skip      10=Back 11=Next 12=Cursor
```

Press PF3

Enter *21*

Panel 52

```
---------------------- Precise Linear System - Hilbert ----------------------
COMMAND ===>

A linear system with a Hilbert matrix will be generated. The right hand side
will be randomly generated. The system can be displayed in browse mode,
altered in edit mode.



Specify the number of unknowns of the Hilbert linear system, 21 is maximum.

   Unknowns ===> 21     (Number of unknowns of the Hilbert linear system)


Specify the relative diameter of the components of the system to be generated.
The relative diameter of an interval is the positive difference of its two
components divided by the maximum of the absolute values of its two components.
Specify 0 for a linear system with exact Hilbert matrix.

   Diameter ===> 0


Press ENTER to generate Hilbert linear system
Press END KEY (PF3) to cancel request                              PF1=Help
```

Press Enter

Enter *i*

Panel 53

```
---------------------- Precise Linear System Solving  HILBERT SYSTEM GENERATED
COMMAND ===> i

The solution of any square linear system is computed. The components of the
solution are displayed rounded to the smallest enclosing intervals.
The results are algorithmically verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random linear system
   H  HILBERT     - Generate linear system with Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise linear system solving


Rounding ===> I      ( N for nearest, I for interval )

If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                   PF1=Help
```

Press Enter

Panel 54

```
---------------------- Precise Linear System - Inclusion ----------------------
COMMAND ===>

The decimal linear system has been converted to the smallest enclosing
hexadecimal interval linear system. Because this system consists of intervals
least significant bit accuracy may not be obtainable. The solution is:

   Component  1   ( -0.2473518797649905D+16 , -0.2473518797649904D+16 )
              2   (  0.1032134946608964D+19 ,  0.1032134946608965D+19 )
              3   ( -0.1071895026985514D+21 , -0.1071895026985513D+21 )
              4   (  0.4901702453207354D+22 ,  0.4901702453207356D+22 )
              5   ( -0.1242977050722214D+24 , -0.1242977050722213D+24 )
              6   (  0.1978314185638057D+25 ,  0.1978314185638059D+25 )
              7   ( -0.2132554688407255D+26 , -0.2132554688407254D+26 )
              8   (  0.1637428856231855D+27 ,  0.1637428856231856D+27 )
              9   ( -0.9271855643310944D+27 , -0.9271855643310941D+27 )
                                              . . .
             20   (  0.3151023559932267D+28 ,  0.3151023559932268D+28 )
             21   ( -0.3143680061512611D+27 , -0.3143680061512609D+27 )
Additionally the non-singularity of every matrix contained in the interval
matrix and the correctness of the displayed results have been algorithmically
verified.
Press ENTER to browse complete result
Press END KEY (PF3) to terminate                                   PF1=Help
```

Press Enter

Panel 55

```
BROWSE - ACRITH OUTPUT A1 ---------------------------- LINE 000000 COL 001 080
COMMAND ===>                                              SCROLL ===> PAGE
****************************************** TOP OF DATA ******************************************
COMPONENT  1:  ( -0.2473518797649905D+16 , -0.2473518797649904D+16 )
           2:  (  0.1032134946608964D+19 ,  0.1032134946608965D+19 )
           3:  ( -0.1071895026985514D+21 , -0.1071895026985513D+21 )
           4:  (  0.4901702453207354D+22 ,  0.4901702453207356D+22 )
           5:  ( -0.1242977050722214D+24 , -0.1242977050722213D+24 )
           6:  (  0.1978314185638057D+25 ,  0.1978314185638059D+25 )
           7:  ( -0.2132554688407255D+26 , -0.2132554688407254D+26 )
           8:  (  0.1637428856231855D+27 ,  0.1637428856231856D+27 )
           9:  ( -0.9271855643310944D+27 , -0.9271855643310941D+27 )
          10:  (  0.3966910448631423D+28 ,  0.3966910448631425D+28 )
          11:  ( -0.1303944613762558D+29 , -0.1303944613762557D+29 )
          12:  (  0.3328300282590987D+29 ,  0.3328300282590989D+29 )
          13:  ( -0.6633261097705267D+29 , -0.6633261097705266D+29 )
          14:  (  0.1032761999720112D+30 ,  0.1032761999720114D+30 )
          15:  ( -0.1250122697736719D+30 , -0.1250122697736718D+30 )
          16:  (  0.1163298479609513D+30 ,  0.1163298479609514D+30 )
          17:  ( -0.8156254515729710D+29 , -0.8156254515729707D+29 )
          18:  (  0.4165647507862696D+29 ,  0.4165647507862697D+29 )
          19:  ( -0.1461931038069801D+29 , -0.1461931038069800D+29 )
          20:  (  0.3151023559932267D+28 ,  0.3151023559932268D+28 )
          21:  ( -0.3143680061512611D+27 , -0.3143680061512609D+27 )
```

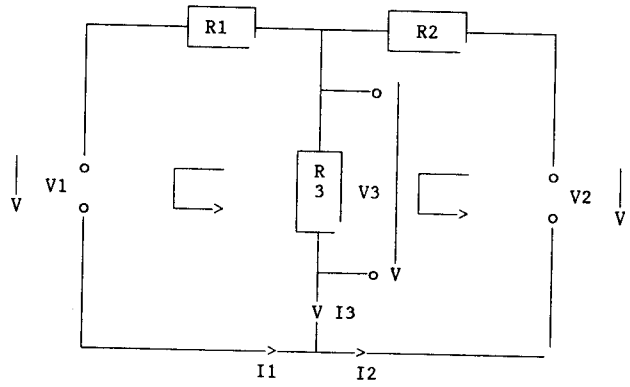
Press PF3

5) Application with Uncertain Data. Data used in engineering problems are usually output of a meter, some material constant, etc. In general, such data are not exactly representable in hexadecimal short or long format. Moreover, many data have an intrinsic error margin. Up to now, it has not been possible to compute with uncertain data. The only way to handle error margins was to solve a problem with several sample values to obtain some 'feeling' about the margins of the solution. In general, it is not possible to solve a problem with all possible combinations of lower and upper bounds of the error margins. If only 10 input data are afflicted with errors, there are over 1000 different possibilities, in case of 20 uncertain input data, over one million.

But even if all combinations of values for the lower and upper bounds of the error margins are inserted and the results for all of them are computed without any error, no valid estimate of the maximum variation of the solution for all possible input data is achieved in general.

As a more practical example, consider the following. Let a simple electrical circuit be given with two power supplies and three resistors, each afflicted with an uncertainty. The power supplies are supposed to maintain the voltage with a tolerance of 2 %, the resistors are of 5 % quality. The problem is to determine the currents (I1,I2,I3) and the voltage-drop V3 at the resistor R3.

The circuit is the following:



Denote the currents through R1, R2, R3 by I1, I2, I3, respectively. Applying Kirchhoff's rules yields the following system of linear equations:

$$
\begin{array}{rcrcrcl}
R1*I1 & & & - & V3 & = & - V1 \\
 & & R2*I2 & + & V3 & = & V2 \\
I1 & - & I2 & + & V3/R3 & = & 0 \\
\hline
 & & & & V3/R3 & = & I3
\end{array}
$$

The values for the components are

V1	5 Volt ± 2 % ,	i.e.	4.9	V1	5.1
V2	10 Volt ± 2 % ,	i.e.	9.8	V2	10.2
R1	1 kOhm ± 5 % ,	i.e.	950	R1	1050
R2	5 kOhm ± 5 % ,	i.e.	4750	R2	5250
R3	10 kOhm ± 5 % ,	i.e.	9500	R3	10500

A common technique is to solve the linear system using the smallest values, the medium values and the maximum values for resistance and voltage. These sets of values are given by

	Smallest	Medium	Largest	
V1	4.9	5.0	5.1	Volts
V2	9.8	10.0	10.2	Volts
R1	950	1000	1050	Ohm
R2	4750	5000	5250	Ohm
R3	9500	10000	10500	Ohm

Note, that the smallest input data don't result in the smallest solution for every variable. The following is a table for the different solutions of the linear system for the different possibilities of input data:

	Smallest	Medium	Largest	
I1	0.396	0.385	0.373	mA
I2	0.952	0.923	0.896	mA
V3	5.27	5.38	5.49	V
I3	0.555	0.538	0.523	mA

The numbers seem to reflect fairly precise values for the different currents with small errors due to the uncertainties of the resistors and the power supplies.

In contrast, the ACRITH Subroutine Library allows to specify uncertainties of constants directly with the problem. That means, for instance, to pass to the algorithm the intervals (950,1050) for R1 and (4750,5250) for R2, etc.

consider a system of linear equations A*x=b, where A is the matrix, x the vector of unknowns and b the right hand side vector. For our example we get the following representation:

$$A = \begin{bmatrix} R1 & 0 & -1 \\ 0 & R2 & 1 \\ 1 & -1 & 1/R3 \end{bmatrix}, \quad x = (I1,I2,V3) , \; b = (-V1,V2,0)$$

ACRITH allows to enter directly uncertain data (tolerances). In our example the interval input data according to the tolerances are the following:

```
     3
(950,1050)        0              -1
     0       (4750,5250)          1
     1            -1       (0.95238D-4,1.0527D-4)
(-5.1,-4.9)   (9.8,10.2)          0
```

In the first line, the dimension of the linear system is entered followed by the matrix data (with tolerances) and, in the last line, the right hand side (again with tolerances). To obtain 1/R3, the interval given for R3 has to be inverted. To obtain the lower (left) bound of 1/R3, the smallest possible value has to calculated. That is 1/10500. Accordingly the right bound of 1/R3 is the inverse of 9500 because this is the largest possible value when inverting any number out of the tolerance (9500,10500).

Using the data of the table above, the problem can directly be solved in the Online Training Component of ACRITH. The result computed by the subroutine of the ACRITH Subroutine Library gives the variation for the different currents due to the tolerances of the input data.

The following table shows the margin of the result computed by ACRITH:

	ACRITH Subroutine Library			
I1	0.26	I1	0.51	mA
I2	0.82	I2	1.03	mA
V3	5.1	V3	5.6	V
I3	0.49	I3	0.59	mA

These are sharp bounds for the solution of the interval linear system. They estimate the range of the voltages and currents for data varying within the given tolerances.

In contrast the method of solving the linear system for lower, medium and upper bounds of input tolerances gives, e.g., an underestimation of the possible current I3 by more than 10 percent.

If the problem is more ill-conditioned these differences may be much larger. The results computed by any algorithm of the ACRITH Subroutine Library provide the additional advantage that every result is automatically verified to be correct.

6) A Conventional Program and an ACRITH Program in Comparison. To solve the problem of the previous example one would traditionally use a Monte Carlo method. In the following the corresponding programs are compared.

In the Monte Carlo program some matrix and some right hand side are randomly generated out of the tolerances. With this data the linear system is solved. These steps, generating data and solving the linear system are performed 20 times, each time computing the minimum and the maximum values for the voltage V3 (i.e. the third component of the solution vector). Finally, the extreme value for I3 are calculated and printed.

Comparing the program for the Monte Carlo method with the program using ACRITH shows, that the latter is much simpler and shorter. Comparing the results of the two approaches shows a 3 % underestimation of the range of I3 by the Monte Carlo method. Moreover, the Monte Carlo program required more than 5 times as much computing time as the ACRITH program measured on a large main frame, an IBM 3081K. On the 4361 with ACRITH in microcode this ratio is even worse.

The new algorithms based on the inclusion theory used in ACRITH perform faster because of the ability to handle data with tolerances directly without having to consider a larger number of examples. Moreover, the ACRITH results are verified to be correct.

MONTE CARLO: The subroutine "SOLVE" is a standard linear system solver, e.g. Gaussian elimination

```
      PROGRAM CIRC
      REAL MATL(3,3),MATR(3,3),BL(3),BR(3),SOLMIN(3),SOLMAX(3)
      REAL MAT(3,3),B(3),SOL(3)
      DATA MATL / 950.0,0.0,-1.0,0.0,4750.0,1.0,1.0,-1.0,9.5238E-5/
      DATA MATR /1050.0,0.0,-1.0,0.0,5250.0,1.0,1.0,-1.0,1.0527E-4/
      DATA BL   / -5.1, 9.8,0.0/
      DATA BR   / -4.9,10.2,0.0/
C
      U = 0.0
      DO 10 I=1,3
         SOLMIN(I) =  1.E70
   10    SOLMAX(I) = -1.E70
C
      DO 30 I=1,20
C
         DO 20 J=1,3
            U = RANDOM(U)
            B(J) = BL(J) + U*(BR(J)-BL(J))
            DO 20 K=1,3
               U = RANDOM(U)
   20          MAT(J,K) = MATL(J,K) + U*(MATR(J,K)-MATL(J,K))
C
         CALL SOLVE(3,MAT,B,SOL,IER)
         IF(IER.NE.0) STOP
C
         DO 30 J=1,3
            SOLMIN(J) = MIN(SOLMIN(J),SOL(J))
   30       SOLMAX(J) = MAX(SOLMAX(J),SOL(J))
C
      D1 = SOLMIN(3)/950.0
      D2 = SOLMIN(3)/1050.0
      D3 = SOLMAX(3)/950.0
      D4 = SOLMAX(3)/1050.0
      DMIN = MIN(D1,D2,D3,D4)
      DMAX = MAX(D1,D2,D3,D4)
C
      WRITE(6,*) DMIN,' <= I3 <= ',DMAX
C
      STOP
      END
```


ACRITH: The ACRITH subroutine "ILSS" solves a linear interval system with high accuracy and verified bounds for the solution

```
      PROGRAM CIRC
      REAL MATL(3,3),MATR(3,3),BL(3),BR(3)
      DATA MATL / 950.0,0.0,-1.0,0.0,4750.0,1.0,1.0,-1.0,9.5238E-5/
      DATA MATR /1050.0,0.0,-1.0,0.0,5250.0,1.0,1.0,-1.0,1.0527E-4/
      DATA BL   / -5.1, 9.8,0.0/
      DATA BR   / -4.9,10.2,0.0/
C
         CALL ILSS(3,MATL,MATR,BL,BR,IER)
         IF(IER.NE.0) STOP
C
      CALL IVDIV(1,BL(3),BR(3),9500.E0,10500.E0,EL,ER,IER)
C
      WRITE(6,*) EL,' <= I3 <= ',ER
C
      STOP
      END
```

RESULTS FOR ELECTRICAL CIRCUIT

Conventional Program (Monte Carlo):

0.499 mA ≤ I3 ≤ 0.576 mA

Computing Time: 26 msec

ACRITH Program:

0.494 mA ≤ I3 ≤ 0.587 mA

Computing Time: 5 msec

7) Polynomial Handling. It might be cumbersome to go back to the main menu from any option every time to enter a new option like polynomial handling. A shortcut is to enter *=p* in the command line of any panel to enter directly the polynomial handling menu panel 57 as shown in the linear system solving panel 56.

For polynomials only point input data is possible. After entering *e* the input polynomial of degree 4 is entered in panel 59. The coefficients are of moderate size, between 1 and 12000. To obtain an overview first a graph is displayed for x-values between -1 and 1.5. There is a zero near -0.7 and at least one zero somewhere near 0.5.

To find out in panel 62 *z* is entered to compute an inclusion of a zero of the polynomial near 0.5. The answer is given in panel 63: there is a zero of the polynomial at 0.7070707070... This fact is verified to be correct.

In panel 64 the value of the polynomial at 0.707070707, left of the zero is computed. It should be negative and an inclusion of the polynomial value of maximum accuracy is displayed in panel 65. The floating-point approximation computed in long format using Horner's scheme (*the* standard algorithm) is also displayed. It is wrong by two orders of magnitude.

In the following panel 67 the value of the polynomial at 0.707070708, i.e. right of the zero is displayed. The inclusion of maximum accuracy shows the positive value, whereas the floating-point approximation is still negative. Therefore when using floating-point arithmetic it cannot be detected that there is a zero of the polynomial because the values left and right of the zero are both negative.

Enter *=p* Panel 56

```
---------- Precise Linear System Solving ----------
COMMAND ===> =p

The solution of any square linear system is computed. The components of the
solution are displayed rounded to the smallest enclosing intervals.
The results are algorithmically verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random linear system
   H  HILBERT     - Generate linear system with Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise linear system solving


Rounding ===> I     ( N for nearest, I for interval )

If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                              PF1=Help
```

Press Enter

Enter *e* Panel 57

```
---------- Precise Polynomial Handling ----------
COMMAND ===> e

Evaluation of a polynomial and computation of the zeros of a polynomial.
All values are algorithmically verified to be correct.

Enter one of the following options in the command line and press ENTER .

   E  ENTER    - Enter and edit input data
   R  RANDOM   - Generate random polynomial
   B  BROWSE   - Browse input data
   V  VALUE    - Evaluate polynomial     (enter value of variable in input field)
   Z  ZERO     - Compute zero of polynomial (enter approximation in input field)
   G  GRAPH    - Show graph of polynomial   (enter range in input fields)


   Input ===>
         ===>


   Input mode ===>       ( Specify F if in case of options "V", "Z", or "G"
                           the input is to be read from the input file)

Press ENTER to process or END KEY (PF3) to terminate                PF1=Help
```

Press Enter

Panel 58

```
---------- Precise Polynomial Handling - Edit ----------
COMMAND ===>

Enter the degree of the polynomial and then its coefficients.

Example                    3         2
                     3.5 x   - 12 x   - 7.3 x  - 56
A sample input file may look as follows:

EDIT --- ACRITH INPUT A ---------------------------------- COLUMNS 001 072
COMMAND ===>                                               SCROLL ===> HALF
****** ******************************** TOP OF DATA ********************************
000001 3
000002 3.5 -12 -7.3 -56
****** ***************************** BOTTOM OF DATA ******************************

Note: Make sure that NUMBER mode is turned off.

The coefficients of the polynomial may be followed by input data which are
processed in the same way as input data from the input fields.
To make use of this feature enter "F" in the mode field in combination with
options "V", "Z" and "G". In this case several input data may be processed in
sequence.
Press ENTER to continue                                             PF1=Help
```

Press Enter

Enter *input* Panel 59

```
EDIT --- ACRITH INPUT A ----------------------------------- COLUMNS 001 072
COMMAND ===>                                                 SCROLL ===> PAGE
****** ******************************** TOP OF DATA ********************************
000001 4
000002 8118 -11482 1 5741 -2030
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
****** ****************************** BOTTOM OF DATA ******************************
```

Press PF3

Enter *g* and *-1* and *1.5* Panel 60

```
---------------------- Precise Polynomial Handling -------------------------
COMMAND ===> g

Evaluation of a polynomial and computation of the zeros of a polynomial.
All values are algorithmically verified to be correct.

Enter one of the following options in the command line and press ENTER .

   E  ENTER    - Enter and edit input data
   R  RANDOM   - Generate random polynomial
   B  BROWSE   - Browse input data
   V  VALUE    - Evaluate polynomial    (enter value of variable in input field)
   Z  ZERO     - Compute zero of polynomial (enter approximation in input field)
   G  GRAPH    - Show graph of polynomial   (enter range in input fields)


   Input ===> -1
         ===> 1.5


   Input mode ===>       ( Specify F if in case of options "V", "Z", or "G"
                           the input is to be read from the input file)

Press ENTER to process or END KEY (PF3) to terminate                 PF1=Help
```

Press Enter

Panel 61

```
------------------ Precise Polynomial Handling - Graph ----------------------
            *
 11103.755+
          I
          I*
          I                                                                *
          I
  7472.529+ *                                                             *
          I                                                              *
          I  *
          I                                                             *
          I   *                                                        *
  3841.304+                                                           *
          I    *                                                     *
          I                                                         *
          I     *                                                 **
          I      *                                             ***
      0.0 +-------*-----------------------------****************-----------
          I        *                          ******
          I         *                     ****
          I          *                ****
          I           ***         ****
 -3421.147+--------------*******-----------+----------------+----------------+
 -1.00000000    -0.37500000     0.25000000      0.87500000      1.50000000
```

Press PF3

Enter *z* and *.5* , erase second input line Panel 62

```
---------------------- Precise Polynomial Handling -------------------------
COMMAND ===> z

Evaluation of a polynomial and computation of the zeros of a polynomial.
All values are algorithmically verified to be correct.

Enter one of the following options in the command line and press ENTER .

   E  ENTER    - Enter and edit input data
   R  RANDOM   - Generate random polynomial
   B  BROWSE   - Browse input data
   V  VALUE    - Evaluate polynomial    (enter value of variable in input field)
   Z  ZERO     - Compute zero of polynomial (enter approximation in input field)
   G  GRAPH    - Show graph of polynomial   (enter range in input fields)


   Input ===> .5
         ===>


   Input mode ===>       ( Specify F if in case of options "V", "Z", or "G"
                           the input is to be read from the input file)

Press ENTER to process or END KEY (PF3) to terminate                 PF1=Help
```

Press Enter

Panel 63

```
------------------ Precise Polynomial Handling - Zeros ----------------------
COMMAND ===>

The coefficients of the input polynomial are exactly representable in
hexadecimal format. A zero has been computed in the following interval:

                    Downwards                    Upwards

            40B50295FAD40A57             40B50295FAD40A58
      (  0.70707070707070057D+00 ,  0.70707070707070072D+00 )

The decimal output numbers in the second line are the converted hexadecimal
numbers rounded downwards and upwards, respectively.

Value of the polynomial at the left  bound: -0.1301535599D-20
                              right bound:  0.1144207120D-21

Additionally the correctness of the displayed results, i.e. the existence of
a zero of the polynomial in the displayed interval has been algorithmically
verified.


Press ENTER to continue
                                                                  PF1=Help
```

Press Enter

Enter *v* and *0.707070707*

Panel 64

```
---------------------- Precise Polynomial Handling --------------------------
COMMAND ===> v

Evaluation of a polynomial and computation of the zeros of a polynomial.
All values are algorithmically verified to be correct.

Enter one of the following options in the command line and press ENTER .

   E  ENTER    - Enter and edit input data
   R  RANDOM   - Generate random polynomial
   B  BROWSE   - Browse input data
   V  VALUE    - Evaluate polynomial     (enter value of variable in input field)
   Z  ZERO     - Compute zero of polynomial (enter approximation in input field)
   G  GRAPH    - Show graph of polynomial   (enter range in input fields)


   Input ===> .707070707
         ===>


   Input mode ===>        ( Specify F if in case of options "V", "Z", or "G"
                            the input is to be read from the input file)

Press ENTER to process or END KEY (PF3) to terminate                PF1=Help
```

Press Enter

Panel 65

```
------------------ Precise Polynomial Handling - Values --------------------
COMMAND ===>

The entered value or the coefficients of the input polynomial are not exactly
representable in hexadecimal format. They have been converted to the nearest
hexadecimal numbers. The polynomial has been evaluated at

   Input was: .707070707

                    Downwards                    Upwards
                B5207D81FEBD857B             B5207D81FEBD857A
      ( -0.721428801531037209D-14 , -0.721428801531037129D-14 )

The decimal output numbers on the second line are the converted hexadecimal
numbers rounded downwards and upwards, respectively.
Additionally the correctness of the displayed results has been
algorithmically verified.

For comparison the result of conventional computation using long format
floating-point operations and Horner's scheme is displayed:
  -0.113686837721616030D-12

Press ENTER to continue
                                                                  PF1=Help
```

Press Enter

Enter *0.707070708*

Panel 66

```
---------------------- Precise Polynomial Handling --------------------------
COMMAND ===> V

Evaluation of a polynomial and computation of the zeros of a polynomial.
All values are algorithmically verified to be correct.

Enter one of the following options in the command line and press ENTER .

   E  ENTER    - Enter and edit input data
   R  RANDOM   - Generate random polynomial
   B  BROWSE   - Browse input data
   V  VALUE    - Evaluate polynomial     (enter value of variable in input field)
   Z  ZERO     - Compute zero of polynomial (enter approximation in input field)
   G  GRAPH    - Show graph of polynomial   (enter range in input fields)


   Input ===> .707070708
         ===>


   Input mode ===>        ( Specify F if in case of options "V", "Z", or "G"
                            the input is to be read from the input file)

Press ENTER to process or END KEY (PF3) to terminate                PF1=Help
```

Press Enter

Enter *=i* Panel 67

```
------------------ Precise Polynomial Handling - Values ---------------------
COMMAND ===> =i

The entered value or the coefficients of the input polynomial are not exactly
representable in hexadecimal format. They have been converted to the nearest
hexadecimal numbers. The polynomial has been evaluated at

   Input was: .707070708

                    Downwards                    Upwards
              361AB0054A5DF3EB             361AB0054A5DF3EC
        (  0.948133331013569525D-13 ,  0.948133331013569652D-13 )

The decimal output numbers on the second line are the converted hexadecimal
numbers rounded downwards and upwards, respectively.
Additionally the correctness of the displayed results has been
algorithmically verified.

For comparison the result of conventional computation using long format
floating-point operations and Horner's scheme is displayed:
  -0.113686837721616030D-12

Press ENTER to continue
                                                                  PF1=Help
```

Press Enter

If a polynomial has to be evaluated successively at many points these can be given all together in the input file after the polynomial coefficients with one evaluation per 'Enter' keystroke.

8) Matrix Inversion. By entering *=i* in the polynomial inclusion panel 67 the matrix inversion option is entered. Here interval and point input data (also mixed) are allowed. A matrix is entered in panel 70 with some components being exact numbers and some components being afflicted with tolerances. The question is whether any combination of input data out of the tolerances is invertible. By entering *i* in panel 71 the point data, i.e. exactly given data, is automatically rounded to the smallest enclosing interval to be sure that any of the specified matrices will be inverted.

The result is displayed in panel 72. It is computationally verified that every matrix out of the tolerances is invertible and the inverse is contained in the displayed bounds. The bounds might appear to be poor. However, the input data has tolerances in the 8th digit, so the inclusion cannot be expected to be of more accuracy. In fact, it can be shown that the computed bounds are sharp.

In panel 75 the input data is slightly altered (the help-menu after entering *e* in panel ... - same as panel ... - has been ommitted). The first digit of the first input number is changed from 5 to 4. Again an inclusion is computed and the solution displayed in panel 77. Now the matrix inversion failed. Indeed there is a singular matrix contained in the input interval matrix. This is a worst case analysis. If there is at least one singular matrix contained in the input interval matrix, no inclusion will be computed. For many problems it is important that they are solvable for *any* combination of input data.

Enter *e* Panel 68

```
------------------------ Precise Matrix Inversion --------------------------
COMMAND ===> e

For any square matrix the inverse is computed and its elements are displayed
rounded to the smallest enclosing intervals. The results are algorithmically
verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random matrix
   H  HILBERT     - Generate Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise inversion

Rounding ===>        ( N for nearest, I for interval )


If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                  PF1=Help
```

Press Enter

Panel 69

```
---------- Precise Matrix Inversion - Edit ----------
COMMAND ===>

Enter the number of rows of the square matrix and then the matrix row by row.
Its components may be real numbers or intervals, (-1,0.2) denotes an inter-
val from -1 to 0.2.

Example                 Matrix   =  1  2.5  (-1,0.2)
                                    0  -12   3.9
                                    4  6.2  -31

A sample input file may look as follows:

EDIT --- ACRITH INPUT A ------------------------------------ COLUMNS 001 072
COMMAND ===>                                                 SCROLL ===> HALF
****** ********************************* TOP OF DATA ********************************
000001 3
000002 1  2.5    (-1,0.2)
000003 0 -12      3.9
000004 4  6.2     -31
****** ******************************** BOTTOM OF DATA ********************************

Note: Make sure that NUMBER mode is turned off.
Press ENTER to continue                                          PF1=Help
```

Press Enter

Enter *input*

Panel 70

```
EDIT --- ACRITH INPUT A ------------------------------------ COLUMNS 001 072
COMMAND ===>                                                 SCROLL ===> PAGE
****** ********************************* TOP OF DATA ********************************
000001 3
000002        5.24754155       (-2.15251107,-2.15251104)     -0.52157255
000003  (8.62972789,8.62972795)      -5.84425364         (1.76003657,1.76003663)
000004 (13.01191425,13.01191434)     -9.53599622                 4.0416457
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
'''''' 
****** ******************************** BOTTOM OF DATA ********************************
```

Press PF3

Enter *i*

Panel 71

```
---------- Precise Matrix Inversion ----------
COMMAND ===> i

For any square matrix the inverse is computed and its elements are displayed
rounded to the smallest enclosing intervals. The results are algorithmically
verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random matrix
   H  HILBERT     - Generate Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise inversion

Rounding ===>        ( N for nearest, I for interval )


If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                 PF1=Help
```

Press Enter

Panel 72

```
---------- Precise Matrix Inversion - Inclusion ----------
COMMAND ===>

The decimal input matrix has been converted to the smallest enclosing
hexadecimal interval matrix. Because this hexadecimal matrix consists of
intervals, least significant bit accuracy may not be obtainable.

      Row  1   (          0.99999984D+00 ,          0.100000022D+01 )
               (         -0.20000006D+01 ,         -0.19999996D+01 )
               (          0.99999983D+00 ,          0.100000032D+01 )

      Row  2   (          0.17518473D+01 ,          0.17518481D+01 )
               (         -0.4094866D+01 ,          -0.4094863D+01 )
               (          0.20092867D+01 ,          0.20092878D+01 )

      Row  3   (           0.9139092D+00 ,           0.9139099D+00 )
               (         -0.3222645D+01 ,          -0.3222642D+01 )
               (          0.17687446D+01 ,          0.17687454D+01 )

Additionally the non-singularity of every matrix contained in the
interval matrix and the correctness of the displayed results have been
algorithmically verified.

Press ENTER to continue                                          PF1=Help
```

Press Enter

Enter *e*

Panel 73

```
------------------------ Precise Matrix Inversion --------------------------
COMMAND ===> e

For any square matrix the inverse is computed and its elements are displayed
rounded to the smallest enclosing intervals. The results are algorithmically
verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random matrix
   H  HILBERT     - Generate Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise inversion

Rounding ===>        ( N for nearest, I for interval )


If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                  PF1=Help
```

Press Enter

Enter *first input digit 4 instead of 5*

Panel 75

```
EDIT --- ACRITH INPUT A1 ------------------------------------ COLUMNS 001 072
COMMAND ===>                                                  SCROLL ===> PAGE
****** ********************************* TOP OF DATA ***********************************
000001 3
000002        4.24754155      (-2.15251107,-2.15251104)    -0.52157255
000003  (8.62972789,8.62972795)     -5.84425364        (1.76003657,1.76003663)
000004 (13.01191425,13.01191434)    -9.53599622              4.0416457
****** ******************************** BOTTOM OF DATA ********************************
```

Press PF3

Enter *i*

Panel 76

```
------------------------ Precise Matrix Inversion --------------------------
COMMAND ===> i

For any square matrix the inverse is computed and its elements are displayed
rounded to the smallest enclosing intervals. The results are algorithmically
verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random matrix
   H  HILBERT     - Generate Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise inversion

Rounding ===>        ( N for nearest, I for interval )


If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                  PF1=Help
```

Press Enter

Enter *=e*

Panel 77

```
------------------ Precise Matrix Inversion - Inclusion --------------------
COMMAND ===> =e

The decimal input matrix has been converted to the smallest enclosing
hexadecimal interval matrix. Because this hexadecimal matrix consists of
intervals, least significant bit accuracy may not be obtainable.




   The matrix inversion failed. The interval matrix probably contains
   a singular matrix.




Press ENTER to continue                                           PF1=Help
```

Press Enter

9) Eigenproblems. Entering *=e* in inversion panel enters the eigenproblem menu 78. Again interval and point input data (also mixed) are allowed. A Hilbert matrix of order 15 is generated in panel 79 and by entering *i* in panel 80 an approximation of the eigenvalues of the matrix is computed. The Hilbert matrix is known to be symmetric and positive definite. Therefore all eigenvalues have to be real and positive.

The approximations of the first 14 eigenvalues are shown in panel 81. By entering *s* in front of the approximated eigenvalue an inclusion of the eigenvalue and corresponding eigenvector can be computed. The inclusion of maximum accuracy for the eigenvalue/eigenvector pair with eigenvalue of largest modulus is shown in panel 82. The approximations computed using a standard floating-point algorithm are correct almost to the last digit. In the next panel the 13th approximation is selected and in panel 84 the corresponding inclusion shown. The inclusion is again of maximum accuracy but now the approximation has only one correct digit. This shows that in one and the same problem large differences of the accuracy of approximations may occur.

In the next panel 85 the 15th eigenvalue is selected. Now the approximation is so bad that no inclusion can be computed. As we saw above, all eigenvalues of the Hilbert matrix are positive, but the approximation is negative.

With ACRITH it is possible to compute an inclusion of the eigenvalue of smallest modulus of the Hilbert matrix. This is an example of chaining several algorithms of ACRITH. The eigenvalue of smallest modulus of the Hilbert matrix is the same as the inverse of the eigenvalue of largest modulus of the inverse Hilbert matrix. By computing first an inclusion of the inverse of the Hilbert 15×15 matrix, second an inclusion of the eigenvalue of largest modulus and third inverting this inclusion yields an inclusion of the eigenvalue of smallest modulus of the original Hilbert 15×15 matrix. This process is described after panel 86.

Enter *h* Panel 78

```
---------------------- Precise Eigenproblem Solving -------------------------
COMMAND ===> h

For any square matrix its eigenvalues and eigenvectors are computed. The
components of the solution are displayed rounded to the smallest enclosing
intervals. The results are algorithmically verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random matrix
   H  HILBERT     - Generate Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise eigenproblem solving


Rounding ===>        ( N for nearest, I for interval )

If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                  PF1=Help
```

Press Enter

Enter *15* Panel 79


```
---------------------- Precise Eigenproblem - Hilbert ----------------------
COMMAND ===>

A Hilbert matrix will be generated. The matrix can be displayed in browse mode
or altered in edit mode.


Specify the number of rows of the Hilbert matrix, 21 is the maximum allowed.

   Rows      ===> 15     (Number of rows of the Hilbert matrix)


Specify the relative diameter of the components of the matrix to be generated.
The relative diameter of an interval is the positive difference of its two
components divided by the maximum of the absolute values of its two components.
Specify 0 for the exact Hilbert matrix.

   Diameter ===> 0


Press ENTER to generate Hilbert matrix
Press END KEY (PF3) to cancel request                             PF1=Help
```

Press Enter

Enter *i* Panel 80

```
---------------------- Precise Eigenproblem Solving  HILBERT MATRIX GENERATED
COMMAND ===> i

For any square matrix its eigenvalues and eigenvectors are computed. The
components of the solution are displayed rounded to the smallest enclosing
intervals. The results are algorithmically verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random matrix
   H  HILBERT     - Generate Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise eigenproblem solving


Rounding ===>        ( N for nearest, I for interval )

If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                  PF1=Help
```

Press Enter

Enter *s* Panel 81

```
------------------------ Eigenvalue Approximations --------------- ROW 1 OF 15
COMMAND ===>                                                  SCROLL ===> PAGE

Approximations for the eigenvalues have been computed using a standard
numerical algorithm. Type "S" in front of the approximated eigenvalue
to compute an inclusion of an eigenvalue/eigenvector pair near the
approximation. The table can be scrolled up and down using PF7 and PF8 .
Press END KEY (PF3) to terminate.                                  PF1=Help

          Select    Number       Approximation
        ---> s          1    0.4299331047249014D+13
        --->            2    0.9936547218636353D+12
        --->            3    0.1332520875854525D+12
        --->            4    0.1313568026562640D+11
        --->            5    0.1016593080465046D+10
        --->            6    0.6313821577904693D+08
        --->            7    0.3171246988718504D+07
        --->            8    0.1287750936359172D+06
        --->            9    0.4199254746819704D+04
        --->           10    0.1084840214474630D+03
        --->           11    0.2171070793215682D+01
        --->           12    0.3247553864139235D-01
        --->           13    0.3321402365925623D-03
        --->           14    0.7252300609465702D-05
```

Press Enter

Panel 82

```
------------------------ Eigenproblem - Inclusions --------------- ROW 1 OF 17
COMMAND ===>                                                  SCROLL ===> PAGE

The decimal input matrix is exactly representable in hexadecimal format. On the
first line an inclusion of the eigenvalue, on the following lines inclusions of
the components of the corresponding eigenvector are displayed. The table can be
scrolled using PF7 and PF8 .
Additionally the existence of an eigenvalue/eigenvector pair within the
displayed bounds has been algorithmically verified.
Press END KEY (PF3) to terminate.                                  PF1=Help

      Approximation                                Inclusion
   0.4299331047249014D+13 (  0.4299331047249016D+13 ,  0.4299331047249018D+13 )

 1 -0.6643596446265620D+00 ( -0.6643596446265623D+00 , -0.6643596446265622D+00 )
 2 -0.4137949331473300D+00 ( -0.4137949331473303D+00 , -0.4137949331473302D+00 )
 3 -0.3126302510605710D+00 ( -0.3126302510605713D+00 , -0.3126302510605712D+00 )
 4 -0.2549709532830490D+00 ( -0.2549709532830496D+00 , -0.2549709532830495D+00 )
 5 -0.2168397174607330D+00 ( -0.2168397174607340D+00 , -0.2168397174607339D+00 )
 6 -0.1894084499107960D+00 ( -0.1894084499107964D+00 , -0.1894084499107963D+00 )
 7 -0.1685687300943880D+00 ( -0.1685687300943888D+00 , -0.1685687300943887D+00 )
 8 -0.1521175798200880D+00 ( -0.1521175798200883D+00 , -0.1521175798200882D+00 )
 9 -0.1387549281285860D+00 ( -0.1387549281285872D+00 , -0.1387549281285871D+00 )
10 -0.1276583608654450D+00 ( -0.1276583608654454D+00 , -0.1276583608654453D+00 )
```

Press PF3

Enter *s* Panel 83

```
------------------------ Eigenvalue Approximations --------------- ROW 1 OF 15
COMMAND ===>                                                  SCROLL ===> PAGE

Approximations for the eigenvalues have been computed using a standard
numerical algorithm. Type "S" in front of the approximated eigenvalue
to compute an inclusion of an eigenvalue/eigenvector pair near the
approximation. The table can be scrolled up and down using PF7 and PF8 .
Press END KEY (PF3) to terminate.                                  PF1=Help

          Select    Number       Approximation
        --->            1    0.4299331047249014D+13
        --->            2    0.9936547218636353D+12
        --->            3    0.1332520875854525D+12
        --->            4    0.1313568026562640D+11
        --->            5    0.1016593080465046D+10
        --->            6    0.6313821577904693D+08
        --->            7    0.3171246988718504D+07
        --->            8    0.1287750936359172D+06
        --->            9    0.4199254746819704D+04
        --->           10    0.1084840214474630D+03
        --->           11    0.2171070793215682D+01
        --->           12    0.3247553864139235D-01
        ---> s         13    0.3321402365925623D-03
        --->           14    0.7252300609465702D-05
```

Press Enter

Panel 84

```
------------------------ Eigenproblem - Inclusions --------------- ROW 1 OF 17
COMMAND ===>                                                  SCROLL ===> PAGE

The decimal input matrix is exactly representable in hexadecimal format. On the
first line an inclusion of the eigenvalue, on the following lines inclusions of
the components of the corresponding eigenvector are displayed. The table can be
scrolled using PF7 and PF8 .
Additionally the existence of an eigenvalue/eigenvector pair within the
displayed bounds has been algorithmically verified.
Press END KEY (PF3) to terminate.                                  PF1=Help

      Approximation                                Inclusion
   0.3321402365925620-03 (  0.3415837419348373D-03 ,  0.3415837419348374D-03 )

 1  0.6619128653389980-07 (  0.6659063765597216D-07 ,  0.6659063765597219D-07 )
 2 -0.8950097898949470-05 ( -0.9002142128888390D-05 , -0.9002142128888389D-05 )
 3  0.2976502881833600-03 (  0.2993118054333081D-03 ,  0.2993118054333083D-03 )
 4 -0.4233032606405360-02 ( -0.4255515571724218D-02 , -0.4255515571724216D-02 )
 5  0.3173005177817420-01 (  0.3188709817020390D-01 ,  0.3188709817020391D-01 )
 6 -0.1376320264143800+00 ( -0.1382334785745924D+00 , -0.1382334785745923D+00 )
 7  0.3541656488185600+00 (  0.3553021688728727D+00 ,  0.3553021688728728D+00 )
 8 -0.5086693966917000+00 ( -0.5086693966916998D+00 , -0.5086693966916997D+00 )
 9  0.2738905255756580+00 (  0.2686472269320604D+00 ,  0.2686472269320605D+00 )
10  0.2430148230079610+00 (  0.2560336645801443D+00 ,  0.2560336645801444D+00 )
```

Press PF3

Enter *s* Panel 85

```
------------------------ Eigenvalue Approximations -------------- ROW 13 OF 15
COMMAND ===>                                                  SCROLL ===> PAGE

Approximations for the eigenvalues have been computed using a standard
numerical algorithm. Type "S" in front of the approximated eigenvalue
to compute an inclusion of an eigenvalue/eigenvector pair near the
approximation. The table can be scrolled up and down using PF7 and PF8 .
Press END KEY (PF3) to terminate.                                  PF1=Help

          Select   Number       Approximation
        --->          13    0.3321402365925623D-03
        --->          14    0.7252300609465702D-05
        ---> s        15   -0.4074167926411605D-05
******************************** BOTTOM OF DATA ********************************
```

Press Enter

Enter *=i* Panel 86

```
------------------------ Eigenproblem - Inclusions --------------------------
COMMAND ===> =i

The decimal input matrix is exactly representable in hexadecimal format.




   Inclusion failed. Either the approximation for the eigenvalue
   is too bad or the problem is ill-conditioned.




Press ENTER to continue                                            PF1=Help
```

Press Enter

We proceed by entering *=i* in panel 86, generating a Hilbert 15×15 matrix in panel 88 and inverting it in panels 89 and 90. In panel 90 *=c* is entered to proceed directly to the copy option. Now the output file containing the inclusion of the inverse of the Hilbert matrix is copied to the input file in panel 91. Then *=e* is entered to proceed to eigenproblems. The input data is now a matrix afflicted with tolerances. After editing the input data to enter the dimension 15 (the help-menu before entering the editor in panel ... - same as panel ... - has been ommitted) *i* is entered in panel 96 to compute approximations of the eigenvalues. In fact approximations of the midpoint matrix are computed. After selecting the first eigenvalue an inclusion of the corresponding eigenvalue/eigenvector pair is computed in panel 98. This inclusion has to be interpreted in the following way. Take any matrix out of the tolerances of the input matrix. Then it has been computationally verified that there exists an eigenvalue of this matrix within the bounds for the eigenvalue and that there exits an eigenvector of this matrix within the bounds for the eigenvector. The true inverse of the Hilbert 15×15 matrix is within the tolerances of the input matrix, therefore an eigenvalue/eigenvector pair of the true inverse of the Hilbert matrix is within the computed eigenvalue/eigenvector bounds displayed in panel 98.

Next the inclusion interval for the eigenvalue has to inverted. By entering *=c* in panel 98 we proceed to the copy option, copy output to input data and proceed to inversion. After preparing the input data in panels 103 and 104 the inclusion is computed in panel 106.

What is shown in panel 106 is an inclusion of the eigenvalue of smallest magnitude of the original Hilbert 15×15 matrix. This inclusion of the eigenvalue is still of maximum accuracy. The floating-point approximation computed by a standard floating-point algorithm displayed as the 15th component in panel 85 is wrong in sign and magnitude.

Enter *h* Panel 87

```
------------------------ Precise Matrix Inversion ---------------------------
COMMAND ===> h

For any square matrix the inverse is computed and its elements are displayed
rounded to the smallest enclosing intervals. The results are algorithmically
verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random matrix
   H  HILBERT     - Generate Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise inversion

Rounding ===>        ( N for nearest, I for interval )


If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                   PF1=Help
```

Press Enter

Enter *15* Panel 88

```
-------------------- Precise Matrix Inversion - Hilbert --------------------
COMMAND ===>

A Hilbert matrix will be generated. The can be displayed in browse mode,
altered in edit mode.



Specify the number of rows of the Hilbert matrix, 21 is the maximum allowed.

   Rows      ===> 15    (Number of rows of the Hilbert matrix)


Specify the relative diameter of the components of the matrix to be generated.
The relative diameter of an interval is the positive difference of its two
components divided by the maximum of the absolute values of its two components.
Specify 0 for the exact Hilbert matrix.

   Diameter ===> 0



Press ENTER to generate Hilbert matrix
Press END KEY (PF3) to cancel request                          PF1=Help
```

Press Enter

Enter *i* Panel 89

```
------------------------ Precise Matrix Inversion -- HILBERT MATRIX GENERATED
COMMAND ===> i

For any square matrix the inverse is computed and its elements are displayed
rounded to the smallest enclosing intervals. The results are algorithmically
verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random matrix
   H  HILBERT     - Generate Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise inversion

Rounding ===>        ( N for nearest, I for interval )


If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                               PF1=Help
```

Press Enter

Enter =c Panel 90

```
------------------- Precise Matrix Inversion - Inclusion --------------------
COMMAND ===>=c

The decimal input matrix is exactly representable in hexadecimal format.
The inverse is:

      Row  1   (  0.9660427129711063D-10 ,  0.9660427129711064D-10 )
               ( -0.1081967838527640D-07 , -0.1081967838527639D-07 )
               (  0.3985248205243470D-06 ,  0.3985248205243471D-06 )
               ( -0.7173446769438248D-05 , -0.7173446769438247D-05 )

                                          . . .

      Row 15   ( -0.8104363636363637D+06 , -0.8104363636363636D+06 )
               (  0.4389863636363636D+06 ,  0.4389863636363637D+06 )
               ( -0.1402678321678322D+06 , -0.1402678321678321D+06 )
               (  0.2003826173826173D+05 ,  0.2003826173826174D+05 )

Additionally the non-singularity of the matrix and the correctness
of the displayed results have been algorithmically verified.


Press ENTER to browse complete result
Press END KEY (PF3) to terminate                               PF1=Help
```

Press Enter

Enter *ACRITH input* Panel 91

```
--------------------------------- Copy To File ---------------------------------
COMMAND ===>

The last created input or output data (from any option) is copied
to the specified dataset.

If output data are copied to the file ACRITH INPUT A, all text which is present
in the output file will be removed.



Copy from    ===> OUTPUT                ( O Output, I Input )

Copy to      ===> ACRITH input          ( File ID: fn ft fm )

Copy options ===> NEW                   ( N New, A Append, R Replace )




Press ENTER to process or END KEY (PF3) to terminate without copy      PF1=Help
```

Press Enter

Enter =e

Panel 92

```
---------- Copy To File ------   OUTPUT COPIED TO INPUT
COMMAND ===> =e

The last created input or output data (from any option) is copied
to the specified dataset.

If output data are copied to the file ACRITH INPUT A, all text which is present
in the output file will be removed.



Copy from    ===> OUTPUT              ( O Output, I Input )

Copy to      ===> ACRITH INPUT        ( File ID: fn ft fm )

Copy options ===> NEW                 ( N New, A Append, R Replace )




Press ENTER to process or END KEY (PF3) to terminate without copy     PF1=Help
```

Press Enter

Enter e

Panel 93

```
---------- Precise Eigenproblem Solving ----------
COMMAND ===> e

For any square matrix its eigenvalues and eigenvectors are computed. The
components of the solution are displayed rounded to the smallest enclosing
intervals. The results are algorithmically verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random matrix
   H  HILBERT     - Generate Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise eigenproblem solving


Rounding ===>        ( N for nearest, I for interval )

If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                      PF1=Help
```

Press Enter

Enter *15* in first line

Panel 95

```
EDIT --- ACRITH INPUT A1 ---------------------------------- COLUMNS 001 072
COMMAND ===>                                                SCROLL ===> PAGE
****** ********************************* TOP OF DATA ***********************************
'''''' 15
000001  (  0.9660427129711063D-10 ,  0.9660427129711064D-10 )
000002  ( -0.1081967838527640D-07 , -0.1081967838527639D-07 )
000003  (  0.3985248205243470D-06 ,  0.3985248205243471D-06 )
000004  ( -0.7173446769438248D-05 , -0.7173446769438247D-05 )
000005  (  0.7496251874062968D-04 ,  0.7496251874062969D-04 )
000006  ( -0.4997501249375313D-03 , -0.4997501249375312D-03 )
000007  (  0.2248875562218890D-02 ,  0.2248875562218891D-02 )
000008  ( -0.7067894624116514D-02 , -0.7067894624116512D-02 )
000009  (  0.1580459770114942D-01 ,  0.1580459770114943D-01 )
000010  ( -0.2528735632183909D-01 , -0.2528735632183908D-01 )
000011  (  0.2873563218390804D-01 ,  0.2873563218390805D-01 )
000012  ( -0.2264019505398816D-01 , -0.2264019505398815D-01 )
000013  (  0.1175548589341692D-01 ,  0.1175548589341693D-01 )
000014  ( -0.3617072582589824D-02 , -0.3617072582589823D-02 )
000015  (  0.4995004995004995D-03 ,  0.4995004995004996D-03 )
000016
000017  ( -0.1081967838527640D-07 , -0.1081967838527639D-07 )
000018  (  0.1615738638867940D-05 ,  0.1615738638867942D-05 )
000019  ( -0.6695216984809031D-04 , -0.6695216984809030D-04 )
000020  (  0.1285481661083333D-02 ,  0.1285481661083334D-02 )
```

Press PF3

Enter *i*

Panel 96

```
---------- Precise Eigenproblem Solving ----------
COMMAND ===> i

For any square matrix its eigenvalues and eigenvectors are computed. The
components of the solution are displayed rounded to the smallest enclosing
intervals. The results are algorithmically verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random matrix
   H  HILBERT     - Generate Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise eigenproblem solving


Rounding ===>        ( N for nearest, I for interval )

If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                      PF1=Help
```

Press Enter

Enter *s* Panel 97

```
------------------------ Eigenvalue Approximations --------------- ROW 1 OF 15
COMMAND ===>                                                 SCROLL ===> PAGE

Approximations for the eigenvalues have been computed using a standard
numerical algorithm. Type "S" in front of the approximated eigenvalue
to compute an inclusion of an eigenvalue/eigenvector pair near the
approximation. The table can be scrolled up and down using PF7 and PF8 .
Press END KEY (PF3) to terminate.                                  PF1=Help

          Select   Number       Approximation
         ---> s         1    0.1422678487513448D+09
         --->           2    0.4434419711961070D+06
         --->           3    0.2927539801326754D+04
         --->           4    0.3079865756146178D+02
         --->           5    0.4605989793624899D+00
         --->           6    0.9217948022431054D-02
         --->           7    0.2381355467172893D-03
         --->           8    0.7763436051843308D-05
         --->           9    0.3149366070910827D-06
         --->          10    0.2065402981802098D-07
         --->          11    0.1133839641954855D-07
         --->          12    0.2892870999803678D-09
         --->          13   -0.5858474046943137D-09
         --->          14   -0.1671833990173176D-08
```

Press Enter

Enter *=c* Panel 98

```
------------------------ Eigenproblem - Inclusions --------------- ROW 1 OF 17
COMMAND ===> =c                                              SCROLL ===> PAGE

The decimal input matrix has been converted to the smallest enclosing hexa-
decimal interval matrix. Because this hexadecimal matrix consists of intervals,
least significant bit accuracy may not be obtainable. On the first line an
inclusion of the eigenvalue, on the following lines inclusions of the compo-
nents of the corresponding eigenvector are displayed. The table can be scrolled
using PF7 and PF8 .
Additionally the existence of an eigenvalue/eigenvector pair within the
displayed bounds has been algorithmically verified.
Press END KEY (PF3) to terminate.                                  PF1=Help

      Approximation                          Inclusion
    0.1422678487513447D+09 (  0.1422678487513447D+09 ,  0.1422678487513449D+09 )

  1 -0.3463310699752080-09 ( -0.3463310262872857D-09 , -0.3463310262872852D-09 )
  2  0.7097449914311990-07 (  0.7097449923415327D-07 ,  0.7097449923415336D-07 )
  3 -0.3614223620036680-05 ( -0.3614223620037194D-05 , -0.3614223620037189D-05 )
  4  0.8044906839941630-04 (  0.8044906839938947D-04 ,  0.8044906839938955D-04 )
  5 -0.9798608739953030-03 ( -0.9798608739952073D-03 , -0.9798608739952062D-03 )
  6  0.7343143936221230-02 (  0.7343143936221255D-02 ,  0.7343143936221262D-02 )
  7 -0.3625995272162420-01 ( -0.3625995272162423D-01 , -0.3625995272162419D-01 )
  8  0.1229364853457810+00 (  0.1229364853457810D+00 ,  0.1229364853457812D+00 )
```

Press Enter

Enter *ACRITH input* Panel 99

```
------------------------------- Copy To File -------------------------------
COMMAND ===>

The last created input or output data (from any option) is copied
to the specified dataset.

If output data are copied to the file ACRITH INPUT A, all text which is present
in the output file will be removed.



Copy from    ===> OUTPUT                ( O Output, I Input )

Copy to      ===> ACRITH input          ( File ID: fn ft fm )

Copy options ===> NEW                   ( N New, A Append, R Replace )



Press ENTER to process or END KEY (PF3) to terminate without copy     PF1=Help
```

Press Enter

Enter *=i* Panel 100

```
------------------------------- Copy To File ------   OUTPUT COPIED TO INPUT
COMMAND ===> =i

The last created input or output data (from any option) is copied
to the specified dataset.

If output data are copied to the file ACRITH INPUT A, all text which is present
in the output file will be removed.



Copy from    ===> OUTPUT                ( O Output, I Input )

Copy to      ===> ACRITH INPUT          ( File ID: fn ft fm )

Copy options ===> NEW                   ( N New, A Append, R Replace )



Press ENTER to process or END KEY (PF3) to terminate without copy     PF1=Help
```

Press Enter

Enter *e* Panel 101

```
---------------------- Precise Matrix Inversion ----------------------------
COMMAND ===> e

For any square matrix the inverse is computed and its elements are displayed
rounded to the smallest enclosing intervals. The results are algorithmically
verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random matrix
   H  HILBERT     - Generate Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise inversion

Rounding ===>        ( N for nearest, I for interval )


If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                  PF1=Help
```

Press Enter

Panel 102

```
---------------------- Precise Matrix Inversion - Edit ---------------------
COMMAND ===>

Enter the number of rows of the square matrix and then the matrix row by row.
Its components may be real numbers or intervals, (-1,0.2) denotes an inter-
val from -1 to 0.2.

Example                 Matrix   =  1  2.5  (-1,0.2)
                                    0  -12   3.9
                                    4  6.2  -31

A sample input file may look as follows:

EDIT --- ACRITH INPUT A ------------------------------------ COLUMNS 001 072
COMMAND ===>                                                 SCROLL ===> HALF
****** ********************************** TOP OF DATA **********************************
000001 3
000002 1  2.5    (-1,0.2)
000003 0 -12      3.9
000004 4  6.2     -31
****** ******************************** BOTTOM OF DATA ********************************

Note: Make sure that NUMBER mode is turned off.
Press ENTER to continue                                           PF1=Help
```

Press Enter

Enter *1* and *line commands* Panel 103

```
EDIT --- ACRITH INPUT A1 ---------------------------------- COLUMNS 001 072
COMMAND ===>                                                 SCROLL ===> PAGE
****** ********************************** TOP OF DATA **********************************
D 0001          APPROXIMATION                                  INCLUSION
000002 1
(27 03       0.142267848751345D+09 (  0.142267848513447D+09 ,  0.14226784875134
D99 04
000005    1 -0.346331069975208D-09 ( -0.3463310262872857D-09 , -0.34633102628728
000006    2  0.7097449991431199D-07 (  0.7097449923415327D-07 ,  0.70974499234153
000007    3 -0.361422362003668D-05 ( -0.3614223620037194D-05 , -0.36142236200371
000008    4  0.804490683994163D-04 (  0.8044906839938947D-04 ,  0.80449068399389
000009    5 -0.979860873995303D-03 ( -0.9798608739952073D-03 , -0.97986087399520
000010    6  0.734314393622123D-02 (  0.7343143936221255D-02 ,  0.73431439362212
000011    7 -0.362599527216242D-01 ( -0.3625995272162423D-01 , -0.36259952721624
000012    8  0.122936485345781D+00 (  0.1229364853457810D+00 ,  0.12293648534578
000013    9 -0.292848132197070D+00 ( -0.2928481321970702D+00 , -0.29284813219706
000014   10  0.494389915177545D+00 (  0.4943899151775450D+00 ,  0.49438991517754
000015   11 -0.588355963055736D+00 ( -0.5883559630557362D+00 , -0.58835596305573
000016   12  0.482563139615067D+00 (  0.4825631396150671D+00 ,  0.48256313961506
000017   13 -0.259571595965375D+00 ( -0.2595715959653749D+00 , -0.25957159596537
000018   14  0.824089435459906D-01 (  0.8240894354599052D-01 ,  0.82408943545990
000019   15 -0.117030285305591D-01 ( -0.1170302853055913D-01 , -0.11703028530559
****** ******************************** BOTTOM OF DATA ********************************
```

Press Enter

Panel 104

```
EDIT --- ACRITH INPUT A1 ---------------------------------- COLUMNS 001 072
COMMAND ===>                                                 SCROLL ===> PAGE
****** ********************************** TOP OF DATA **********************************
000001 1
000002 (  0.1422678487513447D+09 ,  0.1422678487513449d+09 )
****** ******************************** BOTTOM OF DATA ********************************
```

Press PF3

Enter *i* Panel 105

```
---------------------- Precise Matrix Inversion ----------------------------
COMMAND ===> i

For any square matrix the inverse is computed and its elements are displayed
rounded to the smallest enclosing intervals. The results are algorithmically
verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random matrix
   H  HILBERT     - Generate Hilbert matrix
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise inversion

Rounding ===>        ( N for nearest, I for interval )


If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                PF1=Help
```

Press Enter

Enter *=m* Panel 106

```
------------------- Precise Matrix Inversion - Inclusion --------------------
COMMAND ===> =m

The decimal input matrix has been converted to the smallest enclosing
hexadecimal interval matrix. Because this hexadecimal matrix consists of
intervals, least significant bit accuracy may not be obtainable.




      Row  1   (   0.702899501733378D-08 ,   0.702899501733380D-08 )




Additionally the non-singularity of every matrix contained in the
interval matrix and the correctness of the displayed result have been
algorithmically verified.

Press ENTER to continue                                         PF1=Help
```

Press Enter

10) Matrix Multiplication. Entering *=m* in panel 106 proceeds to the matrix multiplication menu in panel 107. Again interval and point input data (also mixed) are possible.

In panel 109 some ill-conditioned input data is entered. The input data, in fact, is a matrix and its inverse so that the product is the identity matrix. The inclusion is computed in panel 111 and all components of the inclusion matrix are displayed in browse mode by pressing enter. The results are of maximum accuracy.

As can be seen the computed left and right bounds are identical. It has been verified that between the left and right bounds there is the true solution. In this case with identical bounds the true solution is verified to be exactly equal to those bounds.

Because of the property, that any dot product and any matrix multiplication as well is performed with maximum accuracy, the result in panel 112 is equivalent to a proof that the two input matrices in panel 109 are inverses of each other.

A similar techniques can be used to prove that all components of the solution of a linear system are rational integers. This is, for example, an important question related to Petri nets. After having solved the linear system with ACRITH the sharp bounds for the solution contain exactly one integer (if this is not the case for one or more components, the answer is already clear). With those integers the residual A*x-b is computed using the precise dot product. Now the integer solution is correct if and only if the computed residual is exactly zero because of the maximum accuracy property of the ACRITH dot product.

Enter *e* Panel 107

```
---------------------- Precise Matrix Multiplication ----------------------
COMMAND ===> e

For any two square matrices their product is computed. The elements of the
product are displayed rounded to the smallest enclosing intervals. These
values are algorithmically verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random matrices
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise matrix product


Rounding ===>        ( N for nearest, I for interval )

If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                PF1=Help
```

Press Enter

Panel 108

```
------------------- Precise Matrix Multiplication - Edit --------------------
COMMAND ===>

Enter the number of rows of the two square matrices and then the two matrices
row by row. Their components may be real numbers or intervals, (-1,0.2) denotes
an interval from -1 to 0.2.

Example                    Matrix A  =  1   (-1,0.2)
                                        -3      0
                           Matrix B  =  0   -12
                                        4.8   9
A sample input file may look as follows:

EDIT --- ACRITH INPUT A ---------------------------------- COLUMNS 001 072
COMMAND ===>                                              SCROLL ===> HALF
****** ********************************* TOP OF DATA ********************************
000001 2
000002 1  (-1,0.2)
000003 -3  0
000004 0  -12
000005 4.8  9
****** ******************************** BOTTOM OF DATA ******************************
Note: Make sure that NUMBER mode is turned off.
Press ENTER to continue                                              PF1=Help
```

Press Enter

Enter *input*

Panel 109

```
EDIT --- ACRITH INPUT A ---------------------------------- COLUMNS 001 072
COMMAND ===>                                              SCROLL ===> PAGE
****** ********************************* TOP OF DATA ********************************
000001 4
000002     1800052652       -129347451      -2545017186       -458762525
000003     3382992055       -243094187      -4783067268       -862190486
000004     4557892356       -327520671      -6444208377      -1161624518
000005     5183043572       -372443327      -7328082850      -1320949490
000006
000007  7938482679644   -11515086485061   9478166610382   -3576027018417
000008     7938482674      -11515086476      9478166602      -3576027015
000009  5613354935108   -8142395739536   6702075883417   -2528632954429
000010     5613354933      -8142395736      6702075880      -2528632953
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
****** ******************************** BOTTOM OF DATA ******************************
```

Press PF3

Enter *i*

Panel 110

```
------------------- Precise Matrix Multiplication ---------------------------
COMMAND ===> i

For any two square matrices their product is computed. The elements of the
product are displayed rounded to the smallest enclosing intervals. These
values are algorithmically verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random matrices
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise matrix product


Rounding ===>        ( N for nearest, I for interval )

If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                     PF1=Help
```

Press Enter

Panel 111

```
---------------- Precise Matrix Multiplication - Inclusion ------------------
COMMAND ===>

The decimal input matrices are exactly representable in hexadecimal format.
Their product is:


     Row  1   (  0.1000000000000000D+01 ,  0.1000000000000000D+01 )
              (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
              (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
              (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )

                                    . . . . .

     Row  4   (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
              (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
              (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
              (  0.1000000000000000D+01 ,  0.1000000000000000D+01 )


Additionally the correctness of the displayed results has been algorithmically
verified.
Press ENTER to browse complete result
Press END KEY (PF3) to terminate                                     PF1=Help
```

Press Enter

Panel 112

```
 BROWSE - ACRITH OUTPUT A1 ------------------------- LINE 000000 COL 001 080
 COMMAND ===>                                          SCROLL ===> PAGE
************************************ TOP OF DATA ************************************
ROW   1 COLUMN  1:  (  0.1000000000000000D+01 ,  0.1000000000000000D+01 )
        COLUMN  2:  (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
        COLUMN  3:  (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
        COLUMN  4:  (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )

ROW   2 COLUMN  1:  (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
        COLUMN  2:  (  0.1000000000000000D+01 ,  0.1000000000000000D+01 )
        COLUMN  3:  (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
        COLUMN  4:  (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )

ROW   3 COLUMN  1:  (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
        COLUMN  2:  (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
        COLUMN  3:  (  0.1000000000000000D+01 ,  0.1000000000000000D+01 )
        COLUMN  4:  (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )

ROW   4 COLUMN  1:  (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
        COLUMN  2:  (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
        COLUMN  3:  (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
        COLUMN  4:  (  0.1000000000000000D+01 ,  0.1000000000000000D+01 )

********************************** BOTTOM OF DATA **********************************
```

Press PF3

Enter =o

Panel 113

```
---------------------- Precise Matrix Multiplication -----------------------
COMMAND ===> =o

For any two square matrices their product is computed. The elements of the
product are displayed rounded to the smallest enclosing intervals. These
values are algorithmically verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random matrices
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise matrix product



Rounding ===>        ( N for nearest, I for interval )

If some input data are not exactly representable in long floating-point
format and option I selected specify whether the data should be rounded to
the nearest floating-numbers or the smallest enclosing intervals.

Press END KEY (PF3) to terminate                                  PF1=Help
```

Press Enter

11) Linear Programming. Entering *=o* in panel 113 proceeds to the linear programming menu in panel 114. Here only point input data are allowed.

In panel 115 a linear programming problem is randomly generated with 11 equations and 15 unknowns. After entering *i* in panel 116 the computed solution is displayed in panel 117. It is automatically verified that the linear programming problem is solvable and that an optimal solution is enclosed in the displayed left and right bounds in panels 117 and 118. The second part of the solution (panel 118) is obtained by pressing PF8 in panel 117. The maximality of the rank of the input matrix is demonstrated and verified as well.

Enter *r*

Panel 114

```
---------------------- Precise Linear Programming ----------------------
COMMAND ===> r

For a linear programming problem the optimal solution is computed. The
components of the solution are displayed rounded to the smallest enclosing
intervals. The results are algorithmically verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random linear programming problem
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise linear programming problem solving



If some input data are not exactly representable in long floating-point
format, they will be rounded to the nearest representable hexadecimal.

Press END KEY (PF3) to terminate                                  PF1=Help
```

Press Enter

Enter *11* and *15*

Panel 115

```
---------------------- Precise Linear Programming ----------------------------
COMMAND ===>

A random linear programming problem will be generated. It can be
displayed in browse mode, altered in edit mode.



Please specify the number of equations and unknowns.

   Rows      ===> 11     (Number of equations of the problem)
   Columns   ===> 15     (Number of unknowns of the problem)









Press ENTER to generate random linear programming problem
Press END KEY (PF3) to cancel request                          PF1=Help
```

Press Enter

Enter *i*

Panel 116

```
---------------------- Precise Linear Programming --- RANDOM DATA GENERATED
COMMAND ===> i

For a linear programming problem the optimal solution is computed. The
components of the solution are displayed rounded to the smallest enclosing
intervals. The results are algorithmically verified to be correct.


Enter one of the following options in the command line and press ENTER .

   E  ENTER       - Enter and edit input data
   R  RANDOM      - Generate random linear programming problem
   B  BROWSE      - Browse input data
   I  INCLUSION   - Apply precise linear programming problem solving




If some input data are not exactly representable in long floating-point
format, they will be rounded to the nearest representable hexadecimal.


Press END KEY (PF3) to terminate                               PF1=Help
```

Press Enter

Enter *half*

Panel 117

```
---------------------- Precise Linear Programming -------------- ROW 1 OF 17
COMMAND ===>                                          SCROLL ===> half

The decimal input linear programming problem is not exactly representable
in hexadecimal long format. It has been converted to the nearest hexadecimal
input data. On the first line an inclusion of the optimal value of the weight
function, on the following lines inclusions of the components of the optimal
solution are displayed. The table can be scrolled using PF7 and PF8 .

Additionally the maximality of the rank of the input matrix and the
existence of an optimal solution within the displayed bounds have been
algorithmically verified.Press END KEY (PF3) to terminate.          PF1=Help

                                Inclusion
          ( -0.8303726879509207D+00 , -0.8303726879509205D+00 )

        1 (  0.2975126398351612D+00 ,  0.2975126398351613D+00 )
        2 (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
        3 (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
        4 (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
        5 (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
        6 (  0.7890682844290101D+00 ,  0.7890682844290102D+00 )
        7 (  0.1725890691603262D-01 ,  0.1725890691603263D-01 )
        8 (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
```

Press PF8

Enter *=f*

Panel 118

```
---------------------- Precise Linear Programming -------------- ROW 10 OF 17
COMMAND ===> =f                                       SCROLL ===> HALF

The decimal input linear programming problem is not exactly representable
in hexadecimal long format. It has been converted to the nearest hexadecimal
input data. On the first line an inclusion of the optimal value of the weight
function, on the following lines inclusions of the components of the optimal
solution are displayed. The table can be scrolled using PF7 and PF8 .

Additionally the maximality of the rank of the input matrix and the
existence of an optimal solution within the displayed bounds have been
algorithmically verified.Press END KEY (PF3) to terminate.          PF1=Help

                                Inclusion
        8 (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
        9 (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
       10 (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
       11 (  0.3512633054143459D-01 ,  0.3512633054143460D-01 )
       12 (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
       13 (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
       14 (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
       15 (  0.0000000000000000D+00 ,  0.0000000000000000D+00 )
********************************* BOTTOM OF DATA ********************************
```

Press Enter

12) Standard Mathematical Functions. By entering *=f* in the inclusion panel 118 of the linear programming option we proceed directly to standard functions. Here 22 functions are implemented for interval input meaning the following. Take any value out of the input interval and compute the value of the standard function. Then the ACRITH function computes the smallest enclosing bounds for all these function values. An example is given for the interval cosine in panel 120. A lower and an upper bound for the set of true values is computed. These bounds are computationally verified to be correct and of maximum accuracy.

The set of standard functions provided for interval input comprises of the four trigonometric functions and their inverse functions, the four hyperbolic functions and their inverse functions, natural and base 10 logarithm, exponential and power function, square and square root function. All those functions are provided in ACRITH in single and double precision, a total number of 44 functions. In the Online Training Component for ACRITH only the double precision versions are accessible.

Enter *icos* Panel 119

```
----------------------- Evaluate Standard Functions -------------------------
COMMAND ===> icos

Enter the number or the name of the function to be evaluated and press ENTER

Trigonometric functions:                Hyperbolic functions:
    1  ISIN/DISIN                           13  ISINH/DISINH
    2  ICOS/DICOS                           14  ICOSH/DICOSH
    3  ITAN/DITAN                           15  ITANH/DITANH
    4  ICOT/DICOT                           16  ICOTH/DICOTH

Inverse trigonometric functions:        Inverse hyperbolic functions:
    5  IASIN/DIASIN                         17  IARSH/DIARSH
    6  IACOS/DIACOS                         18  IARCH/DIARCH
    7  IATAN/DIATAN                         19  IARTH/DIARTH
    8  IACOT/DIACOT                         20  IARCTH/DIACTH

Logarithmic and exponential functions:  Square root and square functions:
    9  ILOG/DILOG                           21  ISQRT/DISQRT
   10  ILG10/DILG10                         22  ISQR/DISQR
   11  IEXP/DIEXP
   12  IPWR/DIPWR

Press END KEY (PF3) to terminate                                   PF1=Help
```

Press Enter

Enter *input* Panel 120

```
---------------------- (D)ICOS - Cosine Function ----------------------------
COMMAND ===>

Precision       ===> DOUBLE  ( S Single, D Double )

Input type      ===> DEC     ( D Decimal, H Hexadecimal )

Decimal input (ignored for HEX)
   Lower bound ===> 3.14159265358979323
   Upper bound ===> 3.14159265358979324

Hexadecimal input for (D)ICOS, input for HEX, output for DEC
   Lower bound ===>
   Upper bound ===>


Hexadecimal result of (D)ICOS


Rounded to smallest enclosing decimal interval



Press ENTER to process or END KEY (PF3) to terminate                PF1=HELP
```

Press Enter

Enter *=a* Panel 121

```
---------------------- (D)ICOS - Cosine Function ----------------------------
COMMAND ===> =a

Precision       ===> DOUBLE  ( S Single, D Double )

Input type      ===> DEC     ( D Decimal, H Hexadecimal )

Decimal input (ignored for HEX)
   Lower bound ===> 3.14159265358979323
   Upper bound ===> 3.14159265358979324

Hexadecimal input for (D)ICOS, input for HEX, output for DEC
   Lower bound ===> 413243F6A8885A30
   Upper bound ===> 413243F6A8885A31

Hexadecimal result of (D)ICOS
              C110000000000000          C0FFFFFFFFFFFFFF

Rounded to smallest enclosing decimal interval
          ( -0.1000000000000000D+01 , -0.9999999999999999D+00 )


Press ENTER to process or END KEY (PF3) to terminate                PF1=HELP
```

Press Enter

(3) Arithmetic Expressions. Entering *=a* in panel 121 proceeds to arithmetic expressions. Here only point input data are allowed. Expressions may consist of +,-,*,/,(,) and **. As long as no over- or underflow occurs bounds of maximum accuracy for the value of the arithmetic expression are computed. Special care has to be taken in case of ** and successive divisions (see [4]).

A simple expression in two variables is entered in panel 122. The values for the variables are entered in panel 123. The true value of the arithmetic expression is computed by ACRITH to be exactly 1, the floating-point approximation computed in long format (17 decimals) is approximately -16000000 (minus sixteen million).

The value of the expression is proved to be exactly 1 because the left and the right bound for the solution in panel 124 are both exactly 1.

To leave the ACRITH Online Training Component *x* is entered in the main menu (as displayed in 126) or *=x* from any intermediate panel.

Enter *formula* Panel 122

```
---------- Arithmetic Expressions ----------
COMMAND ===>

Enter one of the following options in the command line or enter an arithmetic
expression in the 'Input' field(s).

   E - Edit input file
   F - Evaluate expression from input file



Input ===> x**4 - 4*y**2 - 4*y**4
      ===>
      ===>
      ===>
      ===>
  .



Press ENTER to process or  END KEY (PF3) to terminate              PF1=Help
```

Press Enter

Enter *input* Panel 123

```
---------- Arithmetic Expressions - Values ----------
COMMAND ===>

Enter the values for the variables.

X       ===> 665857
Y       ===> 470832









Press ENTER to evaluate expression or END KEY (PF3) to terminate      PF1=Help
```

Press Enter

Panel 124

```
---------- Arithmetic Expressions - Inclusion ----------
COMMAND ===>

An inclusion for the value of the expression has been computed.

Input was: X**4 - 4*Y**2 - 4*Y**4
           :
           :
           :
           :


Result:      (  0.1000000000000000D+01 ,  0.1000000000000000D+01 )









Press ENTER to give new values or END KEY (PF3) to terminate          PF1=Help
```

Press PF3

Panel 125

```
------------------------- Arithmetic Expressions ---------------------------
COMMAND ===>

Enter one of the following options in the command line or enter an arithmetic
expression in the 'Input' field(s).

   E - Edit input file
   F - Evaluate expression from input file



Input ===> X**4 - 4*Y**2 - 4*Y**4
      ===>
      ===>
      ===>
      ===>



Press ENTER to process or  END KEY (PF3) to terminate                PF1=Help
```

Press PF3

Enter *x*

Panel 126

```
----------------- ACRITH Online Training Component (OTC) --------------------
COMMAND ===> x                                              Date     - 85/03/29
                                                            Time     - 23:17
Enter one of the following options and press ENTER .        User     - RUMP
                                                            Terminal- 3279
 T  Tutorial          - Information about ACRITH OTC        PF Keys - 12
 S  Set Parameters    - Set parameters for dialog with OTC
 O  ISPF Parameters   - Specify terminal and user parameters
 R  Rounding          - Rounding of decimal and hexadecimal numbers
 D  Dot Product       - Computation of an accurate scalar product
 M  Matrix Mult.      - Multiplication of matrices
 L  Linear Systems    - Solution of linear systems
 I  Inverse           - Inversion of matrices
 E  Eigenproblems     - Computation of eigenvalues and eigenvectors
 P  Polynomial        - Evaluation and zeros of polynomials
 O  Optimization      - Linear programming optimization
 A  Arithm. Expr.     - Evaluation of arithmetic expressions
 V  Vector Operat.    - Vector operations
 F  Std. Functions    - Evaluation of standard functions
 C  Copy              - Copy last input or output data to a file
 X  Exit              - Terminate ACRITH OTC using exit defaults
 ISPF ISPF/PDF        - Invoke ISPF Program Development Facility

Press END KEY (PF3) to terminate                                     PF1=Help
```

Press Enter

14) References.

[1] Bohlender, G.: Genaue Summation von Gleitpunktzahlen, COMPUTING Suppl. 1, (1977).

[2] Kulisch, U., Miranker,W.L.: Computer Arithmetic in Theory and Practice, ACADEMIC PRESS, New York (1981).

[3] Rump, S.M.: Solving algebraic Problems with High Accuracy, in "A New Approach to Scientific Computation", Edts. U.W. Kulisch and W.L. Miranker, ACADEMIC PRESS, p. 51-120 (1983).

[4] ACRITH, High-Accuracy Arithmetic Subroutine Library, Program Description and User's Guide, IBM Publications, Document Number SC33-6164-1 (1984).

[5] ISPF, Interactive System Productivity Facility, Dialog Management Services, IBM Publications, Document Number SC34-2088-0 (1982).

[6] Rump, S.M.: New Results on Verified Inclusions, to appear in Springer Lecture Notes (1986).