Rational Number Arithmetic including Square Root Handling

Hikaru Samukawa

Shibaura Institute of Technology

1 Preface

Numerical algorithms are usually written in Fortran or C. In these programming languages, real number arithmetic is simulated by floating-point arithmetic. The floating-point arithmetic is accompanied by rounding errors. Algorithms expressed using only four operations, that is, no square roots and/or elementary functions, are programmable with rational number arithmetic. The rational number arithmetic is rounding error free. Examples of these calculations can be seen in programs written by formula manipulation language and/or by decimal BASIC[1]. Recently the computing speed and memory capacity have become sufficient to solve small size numerical problems by rational number arithmetic.

Decimal BASIC cannot merge rational numbers with other types of numerical numbers. We sometimes change a part of numerical programs from double precision to higher precision to investigate of the causes or errors. In such cases, rational number arithmetic will give us a good result. This is our first motivation to develop a new diagnostic check tool which can be embedded into Fortran or C programs to analyze causes of errors written in Fortran or C programs. By using rational number arithmetic, conjugate gradient methods acheive convergence after n times iterations according to the mathematical theories. We have reported that this feature provides good exercises in numerical methods based on mathematics. To be concrete, understanding a set of orthogonal vectors in n-dimensional space through number of iterations[2].

2 Rational number arithmetic

The data structure of rational numbers includes numerator and denominator of rational number in multi-digit integers. Four arithmetic operations are carried out by

- addition and subtraction : $\dfrac{b}{a} \pm \dfrac{d}{c} = \dfrac{bc \pm ad}{ac}$
- multiplication : $\dfrac{b}{a} \cdot \dfrac{d}{c} = \dfrac{bd}{ac}$
- division : $\dfrac{b}{a} \div \dfrac{d}{c} = \dfrac{bc}{ad}$.

Since the product of two n-digit integers becomes $2n$-digit, every multiplication operation doubles number of digits of the numerator and the denominator. Rational number computation divides them to the irreducible form after every operation.

An example of rational number arithmetic can be shown in decimal BASIC, which is used for educational purpose[1]. In Japan, some mathematical textbooks written for senior high school provide program examples by using decimal BASIC in the section of "numerical methods and computer". Decimal BASIC provides five computing modes, i.e., sixteen digits decimal, a thousand digits decimal, binary (double precision floating-point number of Intel IA32 architecture), complex number with sixteen decimal digits, and rational number.

It is easy to make sure of characteristics of the rational number computation by Gaussian elimination program by comparing these modes. To solve a problem of linear equations of $A\boldsymbol{x} = \boldsymbol{b}$, with Hilbert matrix $a_{ij} = \dfrac{1}{i+j-1}$ of order $n = 12$. The answers of binary mode or sixteen digits decimal computations include 30% errors. It becomes worse to more than 100% errors for $n = 13$ problem. By switching to rational number mode, decimal BASIC gives an exact answer. That is to say, if computing power and capacity of memory are available, algorithms that are carried out within rational number field can completely remove the influence of rounding errors.

3 Square root implementation

Since the menu of the numerical algorithms expressed within only rational numbers are too much cramped. It seems natural to extend the scope of applications by adding some features to the rational number arithmetic. The first enhancement should be a capability to handle square root. By adding this feature, algorithms such as Schmidt orthogonalization, Cholesky factorization, etc. can be focused. We discuss the way of handling square root operation in the following steps.

- data structure,
- interval arithmetic.

3.1 Data structure

The basic approach to handle square root is to avoid extracting square root itself as far as possible. We do not keep the square root $\sqrt{r}$ but keep square of square root r, where r is another rational number $r = \frac{f}{e}$. When a vector $\boldsymbol{x}$ is normalized to $\boldsymbol{y} = \sqrt{r}\boldsymbol{x}$, the multiplier is not multiplied but kept in normalization form as r and $\boldsymbol{x}$. In case of a rank-one update operation is followed, $\boldsymbol{y}\boldsymbol{y}^T$ can be computed exactly. This operation appears in deflation operation to obtain the following eigenpairs after the first eigenpair is obtained.

3.2 Interval arithmetic

Since the square root is an irrational number, its final handling should use approximations or interval arithmetic. The self-validating algorithm is based on interval arithmetic which includes the true solution in the interval, and makes the interval be small through iterations[3].

A naive implementation is to express the interval $\left(\frac{b-1}{a}, \frac{b}{a}\right)$ or $\left(\frac{b}{a}, \frac{b+1}{a}\right)$, where $\frac{b}{a}$ is obtained through popular extraction method. But for square root computation, it is better to go through an expansion into continued fraction.

The square root circulates in the expansion into continued fraction. A high precision computation is not needed to obtain the repetend of the recurring expansion. Once the repetend is obtained, the precision of the square root can easily be extended. An example of $\sqrt{3}$, the continued fraction is

$$\sqrt{3} = 1 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{2 + \cdots}}}}. \tag{1}$$

We use a notation

$$\sqrt{3} = [1, 1, 2, 1, 2, 1, 2, \cdots]$$

for the normalized continued fraction. In this case, the repetend is $\overline{1, 2}$. An $n-$th approximation fraction k_n of $\sqrt{3}$ is written as follows

$$\sqrt{3} \fallingdotseq k_n = [a_0, a_1, a_2, a_3, \cdots, a_n],$$

where a_0 is the integer portion of $\sqrt{3}$.

In case of $n = 6$ and $n = 7$, interval $[k_6, k_7]$ includes the true $\sqrt{3} = 1.73205\cdots$, because

$$k_6 = [1, 1, 2, 1, 2, 1, 2] = \frac{71}{41} = 1.\overline{73170},$$
$$k_7 = [1, 1, 2, 1, 2, 1, 2, 1] = \frac{97}{56} = 1.732\overline{142857}.$$

Though both the numerator and the denominator of k_6 and k_7 are two decimal digits, accuracy of this interval has higher precision than that of popular extraction method.

4 Concluding remarks and plans

There exists self-validation approach based upon interval arithmetic between rough floating-point arithmetic and accurate rational number arithmetic. Nowadays these three approaches for arithmetic are segregated, but boundaries will gradually cease to exist with progress of computer evolution.

References

[1] *http://hp.vector.co.jp/authors/VA008683/.*

[2] A Study of Rational Number Arithmetic, *30th JSST Annual Conference (JSST 2011) International Conference in Modeling and Simulation Technology, proc.* (2011).

[3] *http://www.oishi.info.waseda.ac.jp/ oishi/FAQ/FAQ.html.*