Divisionless Computation of Square Roots Through Continued Squaring

DIRAN SARAFYAN, *University of Florida, Gainesville, Fla.*

Consider the recurrence relation

$$x_{i+1} = \frac{x_i^2}{m} + \frac{a}{m} \tag{1}$$

where a and $m \neq 0$ are constant real numbers.

Starting with a chosen real number x_0 and through the use of (1), a sequence $\{x_i\}$, $i = 0, 1, 2, \cdots$, is obtained.

It is known that if this sequence is convergent then it converges to one of the real roots of the quadratic equation

$$x = \frac{x^2}{m} + \frac{a}{m}. \tag{2}$$

These roots are

$$\alpha = \frac{m - \sqrt{m^2 - 4a}}{2} \tag{3}$$

$$\beta = \frac{m + \sqrt{m^2 - 4a}}{2} \tag{4}$$

with the assumption henceforth valid that

$$m^2 - 4a = n > 0, \tag{5}$$

with the case $n = 0$ being dismissed as trivial.

Consider the right-hand member of equation (2) and let

$$f(x) = \frac{x^2}{m} + \frac{a}{m}.$$

A necessary condition for the convergence of the considered sequence is that $|f'(x)| < 1$. This requirement is satisfied here if

$$x_0 \in \left(\frac{-m}{2}, \frac{m}{2}\right) \quad \text{and} \quad m > 0$$

or if

$$x_0 \in \left(\frac{m}{2}, \frac{-m}{2}\right) \quad \text{and} \quad m < 0.$$

Now, since in the first case the interval $\left(\frac{-m}{2}, \frac{m}{2}\right)$ contains only the root α and in the second case the interval $\left(\frac{m}{2}, \frac{-m}{2}\right)$ contains only the root β, it follows from the well-known theory of functional iteration [1] that if $m > 0$ the sequence generated by relation (1) will converge to α and if $m < 0$ it will converge to β.

If x_0 is an estimate or an initial or starting approximation for either one of the roots of (2), then x_i can be considered as the ith approximation to this root. For the sake of simplicity, α, the smaller root corresponding to the case $m > 0$, shall be considered. Then formula (3) can be written

$$x_i \sim \alpha = \frac{m - \sqrt{n}}{2}.$$

The latter in turn yields

$$m - 2\alpha = \sqrt{n} = r \tag{6}$$

$$m - 2x_i \sim \sqrt{n} = r. \tag{7}$$

Letting $m - 2x_i = r_i$, $i = 0, 1, 2, \cdots$, the sequence $\{r_i\}$ is obtained which converges to $r = \sqrt{n} = m - 2\alpha$. The number r_i shall be called the ith approximation for $r = \sqrt{n}$.

Thus this computational method is based on the following steps:

$$m > 0, \qquad -\frac{m}{2} < x_0 < \frac{m}{2}$$

$$a = \frac{m^2 - n}{4} = 0.25(m^2 - n) \tag{8}$$

$$x_i = \frac{x_{i-1}^2}{m} + \frac{a}{m}, \qquad i = 1, 2, 3, \cdots$$

$$r_i = m - 2x_i \sim \sqrt{n} \tag{9}$$

or

$$\sqrt{n} = m - 2\left[\frac{1}{m}\left(\frac{x_0^2}{m} + \frac{a}{m}\right)^2 + \frac{a}{m} \cdots\right]. \tag{10}$$

Obviously, if m is so chosen that its reciprocal is known (such as 2, 4, 50, 100, etc.), then the arithmetical operation of division is completely eliminated from the required operations. In particular, if n is a number close to unity, then, taking $m = 1$ and $x_0 = 0$, formula (10) becomes [2]

$$\sqrt{n} = 1 - 2[(a^2 + a)^2 + a \cdots].$$

It is known that $e_k = \alpha - x_k$, the committed error on the kth iterate of α, is approximately proportional to the kth power of the asymptotic convergence factor [3]. In the present case,

$$e_k = (\alpha - x_0)\left[\frac{2\alpha}{m}\right]^k.$$

It is seen that the smaller $2\alpha/m$ is, the smaller the committed error is. But since on account of formula (3)

$$\frac{2\alpha}{m} = \frac{m - \sqrt{n}}{m},$$

it follows that, the closer m is to $\sqrt{n}$ or the closer m^2 is to n, the smaller $2\alpha/m$ will result and therefore the better will be the obtained approximations.

Assume now that this condition is satisfied. Then using (1) and neglecting a/m we obtain:

$$x_{i+1} \sim \frac{x_i^2}{m}. \tag{11}$$

With the same iterate x_i written as X_i for the purpose of distinction, the Newton-Raphson method would give here

$$X_{i+1} = \frac{X_i^2 - a}{2X_i - m} \sim \frac{x_i^2}{2x_i - m}. \tag{12}$$

Now m is close to r, and x_i is a close approximation to it. Therefore $x_i \sim m$ or

$$2x_i - m \backsim m.$$

The consideration of the latter enables one to write (12) as:

$$X_{i+1} \sim \frac{x_i^2}{m}. \tag{13}$$

A comparison between relations (11) and (13) shows that $x_{i+1} \sim X_{i+1}$. Since the Newton-Raphson method is a second order iteration process, it follows that if m^2 is taken very close to n then the method described herein shall also become almost a second order iteration process.

It is worthwhile mentioning that this conclusion can also be drawn from graphical considerations. In fact, the graphical representation of a functional iteration consists of approaching to an appropriate point (representing a root) on a curve by vertical and horizontal lines, while in the Newton-Raphson process one approaches by tangential lines to the considered root. If a is small, then the horizontal lines fall quite close to the tangential lines.

As far as the choice of the initial approximation x_0 is concerned, as indicated previously, it can be any number belonging to the open interval $\left(\frac{-m}{2}, \frac{m}{2}\right)$. So if one takes $x_0 = 0$ through the relation (1), $x_1 = \frac{a}{m}$ is obtained. Thus, to save time and also for the sake of convenience, one may take as well $x_0 = a/m$ where m is a number with a known reciprocal and also such that its square is as close as possible to n.

Example: Compute $r = \sqrt{82}$. Here $n = 82$; taking $m^2 = 100$ or $m = 10$ one finds:

$$a = 0.25(m^2 - n) = 0.25(18) = 4.5$$

$$x_0 = \frac{a}{m} = 0.1(4.5) = 0.45$$

$$x_1 = \frac{x_0^2}{m} + \frac{a}{m} = 0.1(0.45)^2 + 0.45 = 0.47025$$

$$x_2 = 0.1(0.47025)^2 + 0.45000 = 0.47211$$

$$r_2 = m - 2x_2 = 10 - 0.94422 = 9.05578$$

This is an approximation correct to four significant figures. It is worthwhile noting that if instead of $m^2 = 100$ or $m = 10$ we had taken $m^2 = 81$ or $m = 9$ we would have obtained still better approximations. However, in this case division by 9 will be required in our computations. In electronic computers the arithmetical operation of division can be avoided by the storage of a few reciprocal numbers in the machine.

We shall now derive recurrence relations in iterates r_i and r_{i+1} and free of x_i and a.

From formula (9) is obtained

$$r_{i+1} = m - 2x_{i+1}.$$

The combination of relations (8) and (9) with the latter formula gives the desired recurrence relation:

$$r_{i+1} = \frac{m^2 + n}{2m} - \frac{(m - r_i)^2}{2m} \tag{14a}$$

This formula can also be put into various equivalent forms, such as:

$$r_{i+1} = \frac{0.5}{m}[(m^2 + n) - (m - r_i)^2] \tag{14b}$$

$$r_{i+1} = r_i + \frac{0.5}{m}(n - r_i^2) \tag{14c}$$

As far as the selection of r_0 is concerned, we may proceed as follows. The relation (9) gives

$$r_0 = m - 2x_0, \tag{15}$$

and since $x_0 \in \left(\frac{-m}{2}, \frac{m}{2}\right)$ it follows that $r_0 \in (0, 2m)$, that is, in formulas (14) any positive number smaller than $2m$ can be taken as r_0. However, since for the sake of convenience it has been suggested above that $x_0 = a/m$, then to this selected value of x_0 the relation (15) yields $r_0 = (m^2 + n)/2m$ as a starting approximation. In this case, formula (14a) can be written:

$$\begin{cases} r_0 = \dfrac{m^2 + n}{2m} \\ r_{i+1} = r_0 - \dfrac{(m - r_i)^2}{2m} \end{cases} \tag{16}$$

EXAMPLE. Compute $r = \sqrt{440}$. Through the use of the formulas (16) and with $m^2 = 400$ or $m = 20$, one finds:

$$r_0 = 21, \quad r_1 = 20.97500, \quad r_2 = 20.97623.$$

The latter is an approximation to $r = \sqrt{440}$ correct to 6 figures and with a relative error of about 0.0000027.

Finally, it must be mentioned that, although formulas (14) and (16) have been derived on the assumption that $m > 0$, it can be easily shown that they are also valid when $m < 0$ on condition that in these formulas m is replaced by its absolute value. For example, if $m < 0$ the formula (14a) would become

$$0 < r_0 < 2\,|m|$$

$$r_{i+1} = \frac{m^2 + n}{2\,|m|} - \frac{(|m| - r_i)^2}{2\,|m|}.$$

However, it is evident that this case of $m < 0$ is of no practical value.

REFERENCES

1. APPELL, PAUL, *Analyse Mathématique*, Tome I, p. 165 (Gauthier-Villars, Éditeur, Paris, 1951).
2. SARAFYAN, DIRAN, Program, Association for Computing Machinery, 12th National Meeting, Houston, June 19, 1957.
3. HILDEBRAND, F. B., *Introduction to Numerical Analysis*, pp. 443–445 (McGraw-Hill Book Company, New York, 1956).

A Start at Automatic Storage Assignment

ROBERT L. PATRICK, *Manhattan Beach, California*

Summary. This technique outlines a method whereby equation sets can be ordered in computational order and checked for compatibility. The technique also allows one to note what equations can be computed in parallel (provided one has parallel arithmetic capabilities) or can be considered a logical entity, i.e., segment. Furthermore, the technique will assist one to intelligently allocate high-speed memory (HSM) so that memory is reassigned to other duties as soon as its present duties are fulfilled. Last, the technique appears to be simple and fast to implement.

Assumptions

For the purposes of this discussion, it is assumed that all variables within a procedure can be related through functions. These functions may vary in size, depending on the capabilities of the person performing the definition.

Introduction

The technique presented below tells how to build up a double array. After this array has been built, the rows of the array are then ordered by a trivial processing operation. The ordered array is then inspected and some light is thrown on the problems of automatic segmenting, intelligent storage allocation, and implied flow.

The Method

OPERATION 1:

For every result or set of results, a function is defined. A row of the array is awarded to each function. The columns of the array are as follows:

Column 1 is the name of the function.

Column 2 is the ordered step number (to be filled in by the process).

Columns 3 through n (the last column) are awarded one column for each named quantity (or set) within the entire procedure.

This array is filled out by entering a functional name in the first column and by placing a bit in the column corresponding to every input to the function. This array we will call the input array.

Simultaneously a similar array is built up, called the output array. The rows of the output array correspond to the rows of the input array. For every function in the input array, the results of that function are coded in the output array by placing a bit in the column of the array which corresponds to each result (or set of results).

OPERATION 2:

After the unordered arrays are constructed, a mask is made up. The format of this mask is exactly the same format as the right-hand section of both arrays, i.e., a single bit position exists in the mask for every column position in the arrays. The inputs to the entire procedure are set into the mask as 1's; the remaining positions are initially set to 0. At this same time, the step counter is initialized to 1.

OPERATION 3:

The mask is applied to the first row of the input array. If that row can "see through" the mask, we have determined that each of the inputs to compute that function is available. The function may be deemed computable at this step. The current contents of the step counter are placed in the step column corresponding to the function that has just been determined to be computable. The mask is then applied to the next row and so on until all rows have been tested for computability.

OPERATION 4:

After all rows have been tested for computability, the step counter is increased by 1. The rows of the output array which correspond to those functions which have just been determined to be computable are all *OR*ed together (and also eliminated from the output array). This is now *OR*ed