Babylonian Method of Computing the Square Root: Justifications Based on Fuzzy Techniques and on Computational Complexity

Olga Kosheleva
Department of Mathematics Education
University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
Emails: olgak@utep.edu

***Abstract*—When computing a square root, computers still, in effect, use an iterative algorithm developed by the Babylonians millennia ago. This is a very unusual phenomenon, because for most other computations, better algorithms have been invented – even division is performed, in the computer, by an algorithm which is much more efficient that division methods that we have all learned in school. What is the explanation for the success of the Babylonians' method? One explanation is that this is, in effect, Newton's method, based on the beast ideas from calculus. This explanations works well from the mathematical viewpoint – it explains why this method is so efficient, but since the Babylonians were very far from calculus, it does not explain why this method was invented in the first place. In this paper, we provide two possible explanations for this method's origin. We show that this method naturally emerges from fuzzy techniques, and we also show that it can be explained as (in some reasonable sense) the computationally simplest techniques.**

I. How to Compute the Square Root: an Iterative Formula Going Back to the Ancient Babylonians

How can we compute the square root $x = \sqrt{a}$ of a given number a? Historically the first method for computing the square root was invented by the ancient Babylonians; see, e.g., [2] and references therein. In this "Babylonian" method, we start with an arbitrary positive number x_0, and then apply the following iterative process:

$$x_{n+1} = \frac{1}{2} \cdot \left(x_n + \frac{a}{x_n} \right). \tag{1}$$

To be more precise, Babylonians rarely described their mathematical procedures in algorithmic form: they usually described them by presenting several examples. Only in the Greek mathematics these procedures were reformulated in the general abstract form. This is true for the square root procedure as well: the first person to describe this procedure in general abstract terms was Heron of Alexandria, mostly known as the inventor of the steam engine; see, e.g., [3]. Because of this, the above algorithm is also known as "Heron's method".

II. Properties of the Babylonian Method

If we start with the value x_0 which is already equal to the square root of a, $x_0 = \sqrt{a}$, then, as one can easily check, the next iteration is the exact same value:

$$x_1 = \frac{1}{2} \cdot \left(x_0 + \frac{a}{x_0} \right) = \frac{1}{2} \cdot \left(\sqrt{a} + \frac{a}{\sqrt{a}} \right) = \frac{1}{2} \cdot \left(\sqrt{a} + \sqrt{a} \right) = \sqrt{a}. \tag{2}$$

It has been proven that this method converges: $x_n \to \sqrt{a}$ for all possible starting values x_0. It is intuitively clear that we closer the original approximation x_0 to the square root, the faster the convergence.

Convergence is not so straightforward to prove, but it is straightforward to prove that if the sequence x_n converges to some value x, then this limit value x is indeed the desired square root. Indeed, by tending to a limit $x_n \to x$ and $x_{n+1} \to x$ in the formula (1), we conclude that

$$x = \frac{1}{2} \cdot \left(x + \frac{a}{x} \right). \tag{3}$$

Multiplying both sides of this equality by 2, we get

$$2x = x + \frac{a}{x}. \tag{4}$$

By subtracting x from both sides, we conclude that $x = \frac{a}{x}$. Multiplying both sides of this equality by x, we get $a = x^2$; this is exactly the defining equation of the square root.

III. Babylonian Method: A Numerical Example

The Babylonian method for computing the square root is very efficient. To illustrate how efficient it, let us illustrate it on the example of computing the square root of 2, when $a = 2$.

Let us start with the simplest possible real number $x_0 = 1$. Then, according to the Babylonian algorithm, we compute the next approximation x_1 to $\sqrt{2}$ as

$$x_1 = \frac{1}{2} \cdot \left(x_0 + \frac{a}{x_0} \right) = \frac{1}{2} \cdot \left(1 + \frac{2}{1} \right) = \frac{1}{2} \cdot 3 = 1.5. \tag{5}$$

The next approximation is

$$x_2 = \frac{1}{2} \cdot \left(x_1 + \frac{a}{x_1} \right) = \frac{1}{2} \cdot \left(1.5 + \frac{2}{1.5} \right) =$$

$$\frac{1}{2}\cdot(1.5+1.3333\ldots)=\frac{1}{2}\cdot 2.8333\ldots=1.4166\ldots. \quad (6)$$

So, after two simple iterations, we already get 3 correct decimal digits of $\sqrt{2}=1.41\ldots$

IV. Babylonian Method Is Very Efficient

The above example is typical. In general, the Babylonian method for computing the square root converges very fast.

In fact, it is so efficient that most modern computers use it for computing the square roots. The computers also take advantage of the fact that inside the computer, all the numbers are represented in the binary code. As a result, division by 2 simply means shifting the binary point one digit to the left: just like in the standard decimal code, division by 10 simply means shifting the decimal point one digit to the left, e.g., from 15.2 to 1.52.

V. The Longevity of the Babylonian Method is Very Unusual

The fact that the Babylonian method for computing the square root has been preserved intact and is used in the modern computers is very unusual.

Even for simple arithmetic operations such as division, the traditional numerical procedures that people has used for centuries turned out to be not as efficient as newly designed ones. For example, in most computers, subtraction and operations with negative numbers are not done as we do it, but by using the 2s complement representation; see, e.g., [1]. Similarly, division is not performed the way we do it, but rather by using a special version of Newton's method, etc.

In contrast, the Babylonian method for computing the square root remains widely used. What is the reason for this longevity? How could Babylonians come up with a method which is so efficient?

VI. Newton's Explanation of the Efficiency of the Babylonian Method

Historically the first natural explanation of the efficiency of the Babylonian method was proposed by Isaac Newton. Newton showed that this method is a particular case of a general method for solving non-linear equation, a method that we now call Newton's method.

Specifically, suppose that we want to solve an equation

$$f(x)=0, \quad (7)$$

and we know an approximate solution x_n. How can we find the next iteration?

We assumed that the known value x_n is close to the desired solution x. So, we can describe this solution as $x=x_n+\Delta x$, where the correction $\Delta x \stackrel{\text{def}}{=} x-x_n$ is relatively small. In terms of Δx, the equation (7) takes the form

$$f(x_n+\Delta x)=0. \quad (8)$$

Since Δx is small, we can use the derivative $f'(x_n)$. Specifically, the derivative $f'(x)$ is defined as the limit

$$f'(x_n)=\lim_{h\to 0}\frac{f(x_n+h)-f(x_n)}{h}. \quad (9)$$

The limit means that the smaller h, the closer is the ratio

$$\frac{f(x_n+h)-f(x_n)}{h} \quad (10)$$

to the derivative $f'(x_n)$. Since Δx is small, the ratio

$$\frac{f(x_n+\Delta x)-f(x_n)}{\Delta x} \quad (11)$$

is close to the derivative $f'(x_n)$:

$$f'(x_n)\approx\frac{f(x_n+\Delta x)-f(x_n)}{\Delta x}. \quad (12)$$

We know that $f(x_n+\Delta x)=f(x)=0$; thus, (8) implies that

$$f'(x_n)\approx-\frac{f(x_n)}{\Delta x}. \quad (13)$$

and hence,

$$\Delta x\approx-\frac{f(x_n)}{f'(x_n)}. \quad (14)$$

So, as the next approximation to the root, it is reasonable to take the value $x_{n+1}=x_n+\Delta x$, i.e., the value

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}. \quad (15)$$

Finding the square root $x=\sqrt{a}$ means finding a solution to the equation $x^2-a=0$. This equation has the form $f(x)=0$ for $f(x)=x^2-a$. Substituting this function $f(x)$ into the general formula (15), we get

$$x_{n+1}=x_n-\frac{x_n^2-a}{2\cdot x_n}. \quad (16)$$

Explicitly dividing each term in the right-hand side expression by x_n, we get

$$x_{n+1}=x_n-\left(\frac{x_n}{2}-\frac{a}{2\cdot x_n}\right). \quad (17)$$

Opening parentheses, we get

$$x_{n+1}=x_n-\frac{x_n}{2}+\frac{a}{2\cdot x_n}. \quad (18)$$

Replacing $x_n-\frac{x_n}{2}$ with $\frac{x_n}{2}$, and moving the common divisor 2 outside the sum, we get the Babylonian formula

$$x_{n+1}=\frac{1}{2}\cdot\left(x_n+\frac{a}{x_n}\right). \quad (19)$$

VII. There Should Be a More Elementary Explanation of the Babylonian Formula

Newton's explanation explains why the Babylonian method is so efficient – because it is a particular case of the efficient Newton's method for solving nonlinear equations.

However, Newton's explanation does not explain how this method was invented in the first place, since

- the main ideas of Newton's method are heavily based on calculus, while
- the Babylonians were very far away from discovering calculus ideas.

We therefore need a more elementary explanation of the Babylonian formula. Two such explanations are provided in this paper.

VIII. Explanation Based on Fuzzy Techniques: Main Idea

We are looking for a square root $x = \sqrt{a}$, i.e., for the value for which

$$x = \frac{a}{x}. \tag{20}$$

Instead of the exact value x, we only know an approximate value $x_n \approx x$. Since $x_n \approx x$, we conclude that

$$\frac{a}{x_n} \approx \frac{a}{x}. \tag{21}$$

Because of (20), we can conclude that

$$\frac{a}{x_n} \approx x. \tag{22}$$

Thus, the desired square root x must satisfy two requirements:

- the first requirement is that $x \approx x_n$;
- the second requirement is that $x \approx y_n$ where we denoted

$$y_n \stackrel{\text{def}}{=} \frac{a}{x_n}. \tag{23}$$

Thus, we must find x from the following requirement:

$$(x \approx x_n)\,\&\,(x \approx y_n). \tag{24}$$

Fuzzy logic is a method for formalizing statements like this; see, e.g., [4], [6]. In fuzzy logic, to find the value x from the above requirement, we must follow the following steps:

First, we select the membership function $\mu_{\approx}(z)$ for describing the "approximate" relation $\approx$. After this selection,

- the degree of confidence in a statement $x \approx x_n$ is equal to $\mu_{\approx}(x - x_n)$; and
- the degree of confidence in a statement $x \approx y_n$ is equal to $\mu_{\approx}(x - y_n)$.

Next, we select a t-norm $f_{\&}(d, d')$ to describe the effect of the "and" operator on the corresponding degrees. After this selection, for each possible real number x, the degree $\mu(x)$ to which this number satisfies the above requirement can be computed as

$$\mu(x) = f_{\&}(\mu_{\approx}(x - x_n), \mu_{\approx}(x - y_n)). \tag{25}$$

Finally, we need to select a defuzzification procedure that transforms the membership function $\mu(x)$ into a single most appropriate value. For example, as this x, it is natural to take the value x for which the degree $\mu(x)$ is the largest possible.

IX. Explanation Based on Fuzzy Techniques: Analysis

Let us consider different possible selections of a membership function and of the t-norm.

As an example, let us take a Gaussian membership function to describe "approximate"

$$\mu_{\approx}(z) = \exp\left(-\frac{z^2}{2\sigma^2}\right) \tag{26}$$

for some $\sigma > 0$, and the product as a t-norm:

$$f_{\&}(d, d') = d \cdot d'. \tag{27}$$

In this case, we have

$$\mu(x) = \mu_{\approx}(x - x_n) \cdot \mu_{\approx}(x - y_n) = \exp\left(-\frac{(x - x_n)^2}{2\sigma^2}\right) \cdot \exp\left(-\frac{(x - y_n)^2}{2\sigma^2}\right) = \exp\left(-\frac{(x - x_n)^2 + (x - y_n)^2}{2\sigma^2}\right). \tag{28}$$

Due to monotonicity of the exponential function, this value attains the largest possible value when the expression

$$(x - x_n)^2 + (x - y_n)^2 \tag{29}$$

is the smallest possible. Differentiating this expression with respect to x and equating the derivative to 0, we conclude that $2(x - x_n) + 2(x - y_n) = 0$, i.e., that

$$x = \frac{x_n + y_n}{2}. \tag{30}$$

If for the same Gaussian membership function for "approximate", we choose $f_{\&}(d, d') = \min(d, d')$ as the t-norm, we get a different expression for $\mu(x)$:

$$\mu(x) = \min(\mu_{\approx}(x - x_n), \mu_{\approx}(x - y_n)) = \min\left(\exp\left(-\frac{(x - x_n)^2}{2\sigma^2}\right), \exp\left(-\frac{(x - y_n)^2}{2\sigma^2}\right)\right). \tag{31}$$

Due to monotonicity of the exponential function, this value is equal to

$$\mu(x) = \exp\left(-\frac{\max((x - x_n)^2, (x - y_n)^2)}{2\sigma^2}\right), \tag{32}$$

and this value is the largest when the expression

$$\max((x - x_n)^2, (x - y_n)^2) \tag{33}$$

is the smallest possible. This expression, in its term, can be rewritten as $\max(|x - x_n|^2, |x - y_n|^2)$. Due to the fact that the absolute values are always non-negative and the square function is monotonic on non-negative values, this expression has the form $(\max(|x - x_n|, |x - y_n|))^2$, and its minimum is attained when the simpler expression $\max(|x - x_n|, |x - y_n|)$ is the smallest possible. Let us show that this expression also attains its smallest possible value at the midpoint (32).

Indeed, in geometric terms, the minimized expression $\max(|x - x_n|, |x - y_n|)$ is simply the largest of the distances $|x - x_n|$ and $|x - y_n|$ between the desired point x and the given points x_n and y_n. Due to the triangle inequality, we have $|x - x_n| + |x - y_n| \geq |x_n - y_n|$. Thus, it is not possible that both distances $|x - x_n|$ and $|x - y_n|$ are smaller than $\frac{|x_n - y_n|}{2}$ – because then their sum would be smaller than $|x_n - y_n|$. So, at least one of these distances has to be larger than or equal to $\frac{|x_n - y_n|}{2}$, and therefore, the largest of these distances $\max(|x - x_n|, |x - y_n|)$ is always larger than or equal to $\frac{|x_n - y_n|}{2}$. The only way for this largest distance is to be equal to $\frac{|x_n - y_n|}{2}$ is when both distance $|x - x_n|$ and

$|x - y_n|$ are equal to $\dfrac{|x_n - y_n|}{2}$, i.e., when the desired point x is exactly at a midpoint (32) between the two given points x_n and y_n.

One can show that we get the exact same answer (32) if we use triangular membership functions, symmetric piece-wise quadratic membership functions, different t-norms, etc.

X. Explanation Based on Fuzzy Techniques: Result

Our analysis shows that for many specific selections of the membership function for "approximate" and of the t-norm, we get the same answer:

$$x = \frac{x_n + y_n}{2}.$$

In other words, for many specific selections, as the next approximation x_{n+1} to the square root x, we take exactly the value x_{n+1} from the Babylonian procedure.

Thus, fuzzy techniques indeed explain the selection of the Babylonian method.

XI. The Important Role of Symmetry

The fact that different choices lead to the same result x can be explained by the symmetry of the problem. Indeed, the problem is symmetric with respect to reflection

$$x \to (x_n + y_n) - x \tag{34}$$

that swaps the values x_n and y_n. Thus, if we get the unique solution x, this solution must be invariant with respect to this symmetry – otherwise, the symmetric point would be another solution [5].

This invariance means that $x = (x_n + y_n) - x$ and thus, that $x = \dfrac{x_n + y_n}{2}$.

XII. This Application of Fuzzy Techniques Is Rather Unusual

The fact that fuzzy techniques can be useful is well known [4], [6]. However, usually, fuzzy techniques lead to a good *approximation*: to an ideal control, to an ideal clustering, etc. What is unusual about the Babylonian algorithm is that here, fuzzy techniques lead to *exactly* the correct algorithm.

XIII. Explanation Based on Computational Complexity: Main Claim

In any iterative procedure for computing the square root, once we have the previous approximation x_n, we can use:

- this approximation x_n,
- the value a, and
- (if needed) constants

to compute the next approximation $x_{n+1} = f(x_n, a)$ for an appropriate expression f.

For the iterative method to be successful in computing the square root, the expression $f(x_n, a)$ should satisfy the following natural properties:

- first, if we start with the value x_n which is already the square root $x_n = \sqrt{a}$, then this procedure should not change this value, i.e., we should have $f(\sqrt{a}, a) = \sqrt{a}$;
- second, this iterative method should converge.

In the Babylonian method, the computation of the corresponding expression $f(x_n, a)$ involves three arithmetic operations:

- a division
$$\frac{a}{x_n}; \tag{35}$$
- an addition
$$x_n + \frac{a}{x_n}; \tag{36}$$
and
- a multiplication
$$0.5 \cdot \left(x_n + \frac{a}{x_n}\right). \tag{37}$$

Our claim is that this is the simplest possible operation.

In other words, our claim is that it is not possible to find an expression $f(x_n, a)$ which would be computable in 0, 1, or 2 arithmetic operations.

Let us prove this claim.

XIV. It Is Not Possible to Have 0 Arithmetic Operations

If we are not allowing any arithmetic operations at all, then as $x_{n+1} = f(x_n, a)$, we should return

- either the value x_n,
- or the value a,
- or some constant c.

In all three cases, we do not get any convergence to the square root:

- in the first case, the values x_n remain the same and never converge to $\sqrt{a}$:
$$x_0 = x_1 = x_2 = \ldots = x_n = \ldots; \tag{38}$$
- in the second case, we start with some initial value x_0 and then repeatedly return the values equal to a:
$$x_1 = x_2 = \ldots = x_n = \ldots = a; \tag{39}$$
- in the second case, we start with a value x_0 and then repeatedly return the values equal to the constant c:
$$x_1 = x_2 = \ldots = x_n = \ldots = c. \tag{40}$$

XV. It Is Not Possible to Have 1 Arithmetic Operation

The arithmetic operation is either addition, or subtraction, or multiplication, or division. Let us consider these four cases one by one. All operations involve the values x_n and a and a possible constant(s) c and c'.

For addition, depending on what we add, we get $x_n + x_n$, $x_n + a$, $x_n + c$, $a + a$, $a + c$, and $c + c'$. In all these cases, for $x_n = \sqrt{a}$, the result is different from $\sqrt{a}$. So, the expression

$f(x_n, a)$ involving only one addition does not satisfy the condition $f(\sqrt{a}, a) = \sqrt{a}$.

For subtraction, depending on what we subtract, we get the expressions $x_n - a$, $a - x_n$, $x_n - c$ with $c \neq 0$, $c - x_n$, $a - c$, $c - a$, and $c - c'$ (we dismiss the trivial expressions of the type $a - a = 0$). In all these cases, for $x_n = \sqrt{a}$, the result is different from $\sqrt{a}$. So, the expression $f(x_n, a)$ involving only one subtraction does not satisfy the condition $f(\sqrt{a}, a) = \sqrt{a}$.

For multiplication, depending on what we multiply, we get $x_n \cdot x_n$, $x_n \cdot a$, $x_n \cdot c$, $a \cdot a$, $a \cdot c$, and $c \cdot c'$. In all these cases, for $x_n = \sqrt{a}$, the result is different from $\sqrt{a}$. So, the expression $f(x_n, a)$ involving only one multiplication does not satisfy the condition $f(\sqrt{a}, a) = \sqrt{a}$.

For division, depending on what we divide, we get the expressions $\frac{x_n}{a}$, $\frac{a}{x_n}$, $\frac{x_n}{c}$ with $c \neq 1$, $\frac{c}{x_n}$, $\frac{a}{c}$, $\frac{c}{a}$, and $\frac{c}{c'}$ (we dismiss the trivial expressions of the type $\frac{a}{a} = 1$). In all these cases, except for the case $\frac{a}{x_n}$, for $x_n = \sqrt{a}$, the result is different from $\sqrt{a}$. So, the expression $f(x_n, a)$ corresponding to all these cases does not satisfy the condition $f(\sqrt{a}, a) = \sqrt{a}$.

In the remaining case

$$f(x_n, a) = \frac{a}{x_n}, \tag{41}$$

we do have

$$f(\sqrt{a}, a) = \frac{a}{\sqrt{a}} = \sqrt{a}, \tag{42}$$

but we do not satisfy the second condition: of convergence. Indeed, in this case,

$$x_1 = \frac{a}{x_0}, \tag{43}$$

then

$$x_2 = \frac{a}{x_1} = \frac{a}{a/x_0} = x_0, \tag{44}$$

and then again,

$$x_3 = \frac{a}{x_2} = \frac{a}{x_0} = x_1, \tag{45}$$

etc. So, here, we have

$$a_0 = a_2 = \ldots = a_{2n} = \ldots, \tag{46}$$

$$a_1 = a_3 = \ldots = a_{2n+1} = \ldots \tag{47}$$

and no convergence.

XVI. It Is Not Possible to Have 2 Arithmetic Operations

Similarly, one can prove that it is not possible to have two arithmetic operations. This can be proven by enumerating all possible sequences of arithmetic operations and then checking that for all possible inputs (x_n, a, or c) the resulting expression $f(x_n, a)$:

- either does not satisfy the requirement $f(\sqrt{a}, a) = \sqrt{a}$;
- or does not lead to the convergence to the square root.

For example, if we first add, and then multiply, then we get the expression $(e + e') \cdot e''$. Replacing each of the possible inputs e, e', and e'' with one of the possible values x_n, a, or c, we get all possible expressions $f(x_n, a)$ corresponding to this case.

For example, if we take $e = x_n$ and $e' = e'' = a$, we get the expression $f(x_n, a) = (x_n + a) \cdot a$. This expression clearly does not satisfy the requirement $f(\sqrt{a}, a) = \sqrt{a}$. If we take $e = x_n$, $e' = a$, and $e'' = c$, then we get the expression $f(x_n, a) = (x_n + a) \cdot a$ which also does not satisfy the same requirement.

By considering all possible cases, we can thus prove that no expression with 2 arithmetic operations is indeed possible.

XVII. Possible Pedagogical Use of This Proof

In our opinion, this proof provides a good pedagogical example of a simple easy-to-handle arithmetic problem that is actually not a toy problem at all: it is related to an efficient algorithm for computing the square root.

For each combination of operations and inputs, it is relatively easy to come up with an explanation of why this particular combination will not work. With a sufficiently large number of students in a class, and a sufficient time allocated for this exercise, students can actually check all the possibilities – and thus get a sense of achievement.

Indeed, we have 4 possible operations coming first, 4 possible operations coming second, so we have $4^2 = 16$ possible sequences of operations. So, if we have 16 students in the class, each student can handle one of these combinations. If we have 32 students, then we can divide students into pairs so that each pair of students handles one combination of operations.

For each of the these combinations of operations, we have 3 options for each of the inputs, so totally, we have a manageable number of $3^3 = 27$ possible combinations.

XVIII. Is Babylonian Formula the Only Possible Operation that Requires Three Arithmetic Operations?

We have shown that every iterative procedure for computing the square root requires at least three different arithmetic operations on each iteration. Since the Babylonian procedure requires exactly three operations, it is thus indeed the fastest possible.

The next natural question is: are there other iterative procedures that require three arithmetic operations on every iteration? The answer is No – but this proof requires considering a very large number of possible combinations.

XIX. A More Manageable Proof: the Babylonian Procedure is the Fastest

In the above discussions, to estimate how fast each computation is, we simply counted the number of arithmetic operations. This counting makes sense as a good approximation to the actual computation time, but it implicitly assumes that all arithmetic operations require the exact same computation time. In reality, in the computer, different arithmetic operations

actually require different computation times. Specifically, in the computer (just like in human computations),

- addition and subtraction are the simplest (hence fastest) operations;
- multiplication, in effect, consists of several additions – of the results of multiplying the first number by different digits of the second one; thus, multiplication takes longer than addition or subtraction;
- finally, division, in effect, consists of several multiplications – and thus requires an even longer time than multiplication.

Each iteration of the Babylonian algorithm consists of one division, one multiplication, and one addition. Since division is the most time-consuming of the arithmetic operations, to prove that the Babylonian algorithm is the fastest, we must first prove that no algorithm without division is possible. Indeed, if we only use addition, subtraction, or multiplication, then the resulting expression is a polynomial. Once we have a polynomial $f(x_n, a)$, the requirement $f(\sqrt{a}, a) = \sqrt{a}$ can only be satisfied for a linear function $f(x_n, a) = x_n$ for which there is no convergence.

Thus, each expression must have at least one division, i.e., at least as many as the Babylonian expression. It can still be faster than the Babylonian formula if we have exactly 1 division and no multiplications, just divisions, addition, and subtraction. By enumerating all possibilities, one can conclude that such an expression is impossible.

Thus, every expression must have at least one division, and at least one multiplication. So, it is not possible to have an expression which is faster than the Babylonian one, but it may be potentially possible to have an expression which is exactly as fast as the Babylonian one, i.e., that consists of:

- one division,
- one multiplication, and
- one addition or subtraction.

Again, one can enumerate all possible combinations of these three operations and see that the Babylonian expression is the only one that leads to convergence to the square root.

ACKNOWLEDGMENTS

The author is thankful to the anonymous referees for valuable suggestions.

REFERENCES

[1] R. E. Bryant and D. R. O'Hallaron, *Computer Systems: A Programmer's Perspective*, Prentice Hall, Upper Saddle River, New Jersey, 2003.
[2] D. Flannery, *The Square Root of Two*, Springer Verlag, 2005.
[3] T. Heath, *A History of Greek Mathematics*, Clarendon Press, Oxford, 1921, Vol. 2.
[4] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic: theory and applications.* Prentice Hall, Upper Saddle River, New Jersey, 1995.
[5] H. T. Nguyen and V. Kreinovich, *Applications of continuous mathematics to computer science*, Kluwer, Dordrecht, 1997.
[6] H. T. Nguyen and E. A. Walker, *A first course in fuzzy logic*, CRC Press, Boca Raton, Florida, 2005.