Unveiling Ancient Algorithms: Examining the Babylonian Technique for Computing Square Roots

Hamzeh Al Masri
Computer Engineering Department
University of Jordan
Amman, Jordan
hmz0197900@ju.edu.jo

***Abstract*— The Babylonian approach, commonly referred to as Heron's approach, is a traditional iterative method for estimating square roots. The historical context and underlying mathematical ideas of the Babylonian Method are analyzed in this paper. We intend to demonstrate the algorithm's effectiveness and study its convergence qualities by implementing it within modern computing environments, like MATLAB.**

The Babylonian Method iteratively improves an initial prediction for the square root of a given number. We examine the detailed computing procedure involved, emphasizing the formula used to update the guess at each iteration. Additionally, we investigate the convergence criterion that establishes whether the technique has accurately approximated the square root.

Keywords*— Babylonian Method, square root approximation, convergence, iterative algorithm, MATLAB.

I. Introduction

Many historical algorithms continue to have an impact on contemporary computer techniques, making mathematics a veritable gold mine of ancient knowledge. One of them is the Babylonian Method, also known as Heron's Method, which is a great example of ancient civilizations' creativity. Considering the primitive computing capabilities at the time, this iterative process, which was developed over 4,000 years ago, offers a way to approximation square roots with astonishing precision.

In this research, we explore the mathematical foundations and historical importance of the Babylonian Method. We seek to shed light on the astounding accomplishments of ancient mathematical minds and demonstrate the continuing importance of their contributions by investigating the algorithm's step-by-step computing process and looking at its convergence features.

The Babylonian Method is based on the idea that an initial approximation for the square root may be improved through a series of iterative processes. The approximation gradually becomes better, and eventually it converges to a more precise result. This strategy is an early illustration of an iterative algorithm, where each iteration uses the earlier estimate to improve the current one.

II. Algorithm

This section explains how this algorithm works and provides the equations that represents the algorithm .

A. *Methodology*

The Babylonian Method begins with a first guess, x0, for the provided number's square root. Using the Babylonian formula, which averages the current guess with the number divided by the guess, it iteratively computes new guesses, x1. The method iterates n times until the convergence criterion is satisfied, that is, until the tolerance is reached between any two successive estimates. Then it outputs the approximate square root.

The accuracy of the calculated square root and the convergence rate of the Babylonian Method can both be influenced by the initial assumption made. Faster convergence and a more accurate result can arise from a solid first approximation that is somewhat near to the real square root. On the other hand, a bad initial guess could need more repetitions to attain convergence and might produce a less accurate approximation.

The method usually converges quickly when the first guess is quite near to the real square root. The Babylonian Method generates a new guess for each iteration by averaging the previous guess and the number divided by the prior guess. This technique of averaging aids in the estimation's improvement and takes it nearer to the actual square root. The difference between the initial estimate and the computed square root at each iteration gets lower the closer the initial guess is to the real square root, which causes convergence to happen more quickly. On the other hand, if the initial guess is far from the true square root, the method will need more iterations to converge. Each iteration takes the estimate closer to the actual square root, but if the initial guess was inaccurate, it can take several iterations to obtain an acceptable degree of precision. In these situations, the method steadily improves the estimate with each iteration, but it can need more computing steps to reach the needed precision.

The kind of the problem and the facts at hand determine what initial assumption is suitable. It can be advantageous to use those figures as the first guess if there have been previous square root estimations or boundaries. However, in the absence of any precise information, it is usual practice to begin with a fair value depending on the size of the number being rooted. Investigating the effects of various initial hypotheses on the algorithm's convergence behavior and accuracy may be done through experimentation.

The speed of convergence of Heron's approach to determine the square root of 100 for various initial assumptions is shown in Fig. 1. Positive and negative estimates converge to the positive and negative roots, respectively. All approximations are overestimated, and numbers closer to the root converge more quickly. So we can conclude that the starting guess and the desired level of accuracy will determine how many iterations are necessary for convergence.

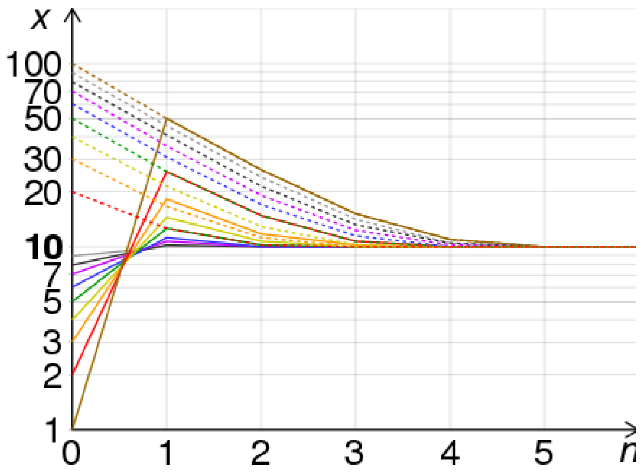


Fig.1. The speed of convergence of the method.[1]

B. *Equations*

By utilizing the initial number for which we wish to get the square root and the previously calculated estimate , X n , the Babylonian formula determines the new guess, designated as X n+1 :

$$X_{n+1} = \frac{X_n + \frac{number}{X_n}}{2}$$

The initial number is represented by "number" in this equation . The current estimate at the n-th iteration is X n . Xn+1 is the updated guess for the (n+1)-th iteration. The calculation in the formula determines the ratio of the number divided by the prior guess as well as the average of the prior guess. The estimate is improved and gets closer to the true square root through this averaging process.

When the method has sufficiently approximated the square root, it meets the convergence requirement. It determines if the range between two successive estimates, X n+1 and X n is less than a specified tolerance value :

$$abs(x_{n+1} - x_n) < \text{Tolerance}$$

The required degree of precision is represented by the tolerance. The algorithm is said to have converged and the approximation can be considered acceptable if the difference between two successive estimates is less than the tolerance.[2]

III. Numeric Examples

- Example-1: Computing the square root for 25 with tolerance set to 0.01.

Assuming X_0=6.25 :
For the first iteration:

1. Current guess (X_0) =6.25
2. $X_1 = \frac{X_0+\frac{25}{X_0}}{2} = \frac{6.25+\frac{25}{6.25}}{2} = 5.125$
3. Check the convergence condition: $abs(x_1 - x_0) = abs$(5.125-6.25)=1.125
4. Enters a new iteration as the value is greater than the tolerance.

For the second iteration:

1. Current guess (X_0) =5.125
2. $X_1 = \frac{X_0+\frac{25}{X_0}}{2} = \frac{5.125+\frac{25}{5.125}}{2} \approx 5.0015$
3. Check the convergence condition: $abs(x_1 - x_0) = abs$(5.0015-5.125) ≈ 0.1235
4. Enters a new iteration as the value is greater than the tolerance.

For the third iteration:

1. Current guess (X_0) ≈ 5.0015
2. $X_1 = \frac{X_0+\frac{25}{X_0}}{2} = \frac{5.0015+\frac{25}{5.0015}}{2} \approx 5.0000$
3. Check the convergence condition: $abs(x_1 - x_0) = abs$(5.0000-5.0015) ≈ 0.0015
4. The absolute difference is now less than the tolerance , so we can end the iterating. Consequently, the estimated square root of 25 is approximately 5.0000.

- Example-2: Computing the square root for 11 with tolerance set to 0.01.

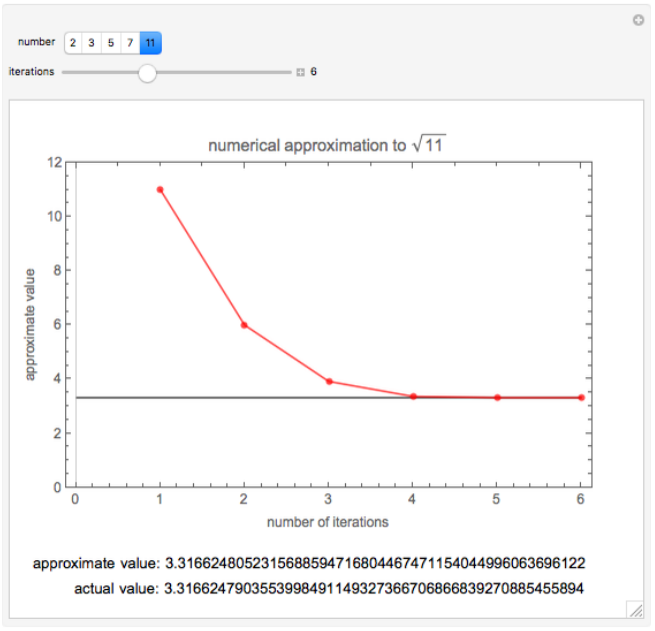
Assuming X_0=3 , this time the assumption is close to the real answer :
For the first iteration:

1. Current guess (X_0) =3
2. $X_1 = \frac{X_0+\frac{11}{X_0}}{2} = \frac{3+\frac{11}{3}}{2} \approx 3.33333$
3. Check the convergence condition: $abs(x_1 - x_0) = abs$(3.33333-3) =0.33333
4. Enters a new iteration as the value is greater than the tolerance.

For the second iteration:

1. Current guess (X_0) ≈ 3.33333
2. $X_1 = \frac{X_0+\frac{11}{X_0}}{2} = \frac{3.33333+\frac{11}{3.33333}}{2} \approx 3.31666$

3. Check the convergence condition: $abs(x_1 - x_0) = abs$(3.31666-3.33333) =0.01667
4. The absolute difference is now less than the tolerance , so we can end the iterating. Consequently, the estimated square root of 11 is approximately 3.31666.


number 2 3 5 7 11
iterations 6

approximate value: 3.3166248052315688594716804467471154044996063696122
actual value: 3.3166247903553998491149327366706866839270885455894

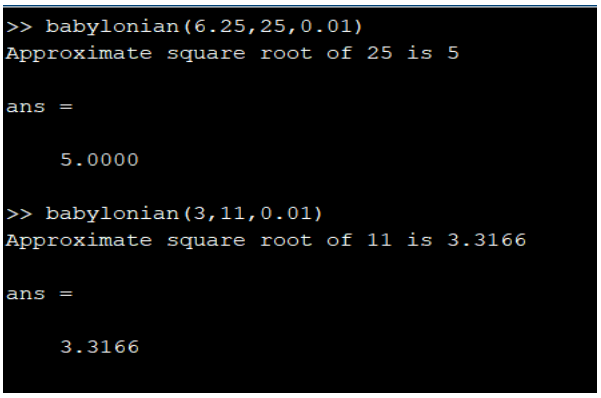
Fig.2. The convergence speed for estimating the square root for the number 11.[3]

IV. MATLAB Implementation

In this code, we build a function called babylonian that accepts three arguments: the number for which we want to compute the square root ‘s’, ‘initial_guess’ as the initial guess, and ‘tol’ as the desired level of accuracy. Them the function returns the approximated root for the number.

```
function sqrt = babylonian(initial_guess,s, tol)

%   sqrt = babylonian(initial_guess,s, tol) returns the
approximate square root of the number.
%   initial_guess: The initial guess for the square root.
%   s: The number whose square root has to be calculated.
%   tol: represents the tolerance

% assigning the initial guess
x0 = initial_guess;

% Iterating till convergence
while true
    % Calculating new guess with the Babylonian method
    x1 = (x0+(s/x0))/2;

    % Checking the convergence condition
    if  tol>abs(x1 - x0)
        break;
    end

    % changing the guess for the following loop
    x0=x1;
end

% Returning the approximate square root
sqrt=x1;
disp(['Approximate square root of ', num2str(s), ' is ',
num2str(sqrt)]);
end
```

Fig.3 shows that the MATLAB function gives the same approximations for the examples solved above.



```
>> babylonian(6.25,25,0.01)
Approximate square root of 25 is 5

ans =

    5.0000

>> babylonian(3,11,0.01)
Approximate square root of 11 is 3.3166

ans =

    3.3166
```

Fig.3. MATLAB code test.

V. Conclusion

A trustworthy and effective approach for estimating square roots is the Babylonian Method. The approach gradually improves the estimate until the required level of accuracy is reached by iteratively updating a guess using the Babylonian formula and testing the convergence criterion.

References

[1] *Methods of computing square roots - Wikipedia*. (2012, January 1). Methods of Computing Square Roots - Wikipedia. https://en.wikipedia.org/wiki/Methods_of_computing_square_roots

[2] Wicklin, R. (2016, May 16). *The Babylonian method for finding square roots by hand*. The DO Loop. https://blogs.sas.com/content/iml/2016/05/16/babylonian-square-roots.html

[3] *Babylonian Algorithm for Computing Square Roots - Wolfram Demonstrations Project*. (n.d.). Babylonian Algorithm for Computing Square Roots - Wolfram Demonstrations Project. http://demonstrations.wolfram.com/BabylonianAlgorithmForComputingSquareRoots/