Comparison and Recommendation Report:

Choosing a Memory- and Time-Efficient Software Implementation to Calculate an Arctangent

Andrew Smith
ECE 2031, Digital Design Lab
Section L00

Georgia Institute of Technology
School of Electrical and Computer Engineering

Submitted
February 23, 2015

Executive Summary

For hardware that cannot calculate trigonometric functions, a software implementation must be used. The purpose of this report is to compare the memory footprints and calculation times of three proposed software implementations to calculate arctangent. The three methods are a Taylor series, a lookup table without interpolation, and a lookup table with interpolation. Calculations must be performed over the domain [0, 1] to within a margin of error of ±1°. The implementation must run in <1 second on a Simple Computer (SCOMP) processor with a 1024 word memory size. The processor will likely contain other code, however, so an ideal implementation will use only a fraction of the 1024 words available.

To meet the 1° accuracy constraint, it was found that a Taylor series would require 15 terms. This requires a memory footprint of 200 words, and a calculation time of 44ms. A lookup table without interpolation requires 23 entries to meet the accuracy constraint. This takes up 53 words of memory, and has an insignificant calculation time. A lookup table with interpolation is accurate to within 1° with three entries. This implementation takes up 63 words of memory and has a calculated time of 2ms.

It was determined that a lookup table without interpolation would be the most effective way to implement an arctangent calculation on SCOMP, due to it having the smallest memory footprint, and the quickest calculation time.

Introduction

This paper examines three different software implementations to calculate an arctangent, and compares them based on calculation time and memory footprint. Calculating an arctangent is an important process for a robot that is attempting to travel from point A to point B in a straight line. The DE2Bot is a simple wheeled robot with a processor called Simple Computer (SCOMP), that is unable to calculate arctangents in hardware. Thus, in order to perform an arctangent calculation, a software solution must be implemented. The constraints for the software implementation are as follows: it must run on SCOMP, which has a memory size of 1024 words, in <1 second, and within a 1° margin of error. For all proposed implementations, only input values from [0, 1) need to be considered, as all other inputs can be calculated using symmetry. There are three main candidates for implementation that must be considered: a Taylor series expansion, a large pre-calculated lookup table, and a reduced lookup table with interpolation between the entries.

A Taylor series is a way of representing a function as an infinite sum of polynomial terms. Taylor series can represent various trigonometric functions "centered" at certain values. For the purpose of this paper, the Taylor series for arctangent centered at zero (in radians) is considered:

$$\arctan(x) \approx x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \ldots \qquad \text{[Eq 1]}$$

The *n*th term of the series can be computed as follows:

$$a_n = (-1)^{(n-1)} \frac{x^{(2n-1)}}{2n-1} \qquad \text{[Eq 2]}$$

When only considering the first *n* terms, the series is only an approximation of the function. The more terms of a Taylor series that you consider, the closer the approximation is to the actual function. Additionally, the error between the Taylor series and the actual function is smallest near the center (in this case zero), and increases the further away from the center one goes, as shown in Figure 1.

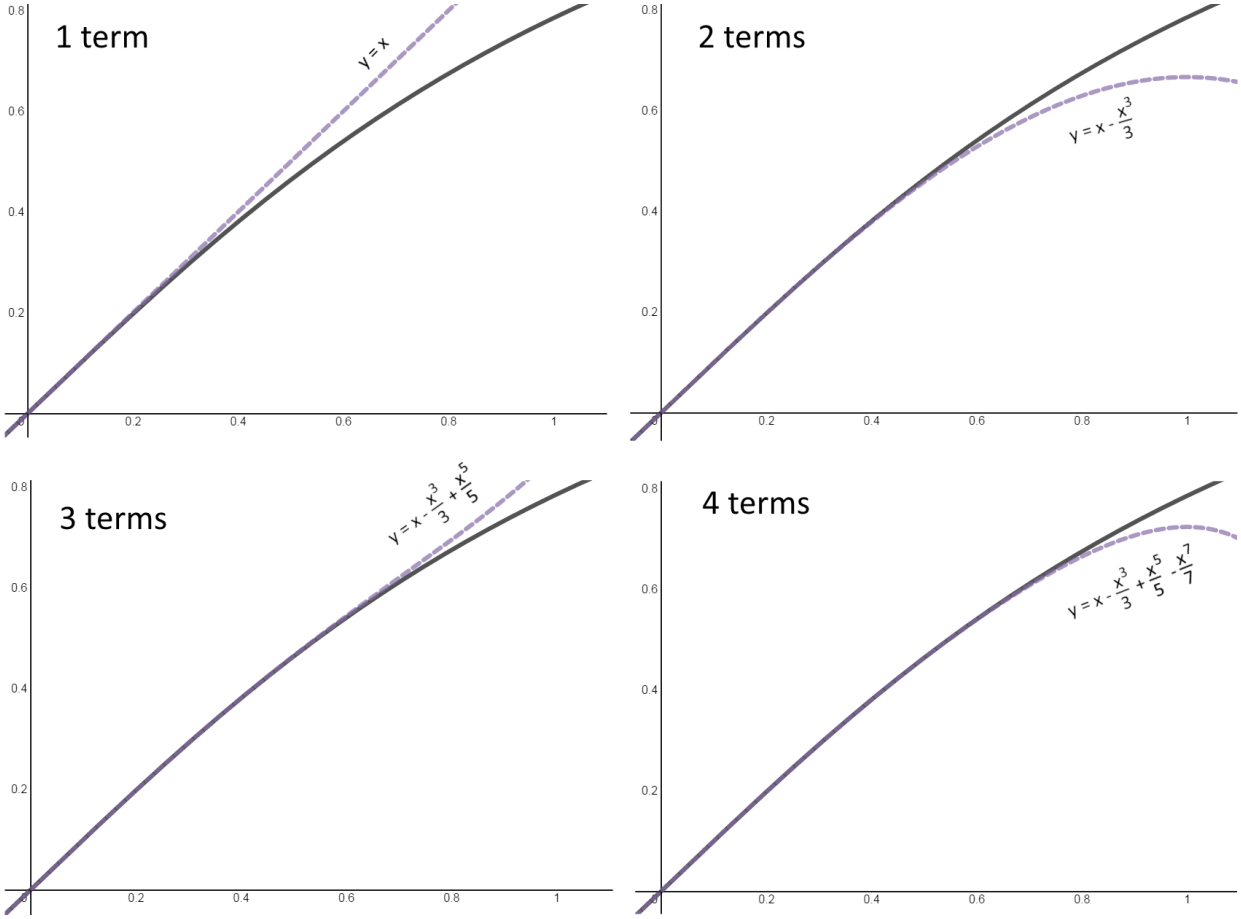

Figure 1. Four graphs, each showing arctan(x) (solid line) and either one, two, three, or four terms of the Taylor series polynomial (dotted line). Note that the error between the two values is always greatest furthest away from the center (zero).

A pre-calculated lookup table without interpolation simply consists of a table of indices and pre-calculated outputs. When the arctangent of a value is to be calculated, the input is rounded to the nearest table index, and the output from that index is the result.

A pre-calculated lookup table with interpolation is similar to one without interpolation, with a small difference. Instead of rounding an input to the nearest index, an input (assuming there are no indices equal to it) will look the closest index less than it and the closest index greater than it. It will then assume that there is a constant rate of change between the outputs of the two indices, and will calculate what its result should be accordingly. This process is known as linear interpolation.

Comparison and Analysis

Taylor Series

To find the most efficient implementation of calculating an arctangent with a Taylor series, it must first be determined how many terms of the Taylor series are needed to guarantee accuracy within ±1 degree. Equation 1 is the Taylor series representation of arctangent (in radians). As stated in the introduction, the error for the Taylor series representing the arctangent function will be greater for values further from zero. This means that in the domain of interest, [0, 1), the error will be greatest when $x = 1$. By ensuring that the worst-case error is <1°, it is guaranteed that all other cases will have error that is <1°. Now, it is just a matter of determining how many terms of the Taylor series are needed so that the value at $x = 1$ is between 44° and 46° (the true value of arctangent at $x = 1$ is 45°). Through trial and error, it can be determined that a Taylor series with 15 terms, and no less, will be accurate to within 1° of the value of arctangent when x = 1.

$$\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots - \frac{1}{27} + \frac{1}{29}\right) \cdot \left(\frac{180}{\pi}\right) = 45.954° \qquad \text{[Eq 3]}$$

Note in (3) that the value of the Taylor series must be converted to degrees, as the implementation of interest must give the result in degrees. A Taylor series software implementation requires and overhead of 50 words of memory, as well as 10 words for each term in the series. Thus, a Taylor series implementation with 15 terms will have a memory footprint of 200 words.

Now timing must be considered. Each multiplication takes 1ms for SCOMP to perform, as well as each division. Powers must be calculated as successive multiplications, but previous calculations can be re-used when calculating even greater powers. For example, the second term of the series must calculate x^3, which takes three multiplications. The third term must calculate x^5. This on its own would take five multiplications, but since the value of x^3 has already been computed, it only needs to take an additional two calculations. The greatest power that must be calculated in the Taylor series with 15 terms is x^{29}, which will take 29 multiplications. However, the calculations for all lesser powers of x are also calculated along the way. In addition to the 29 multiplications, each term (except for the first) must

calculate a division. That makes 14 divisions. Lastly, one additional multiplication must be made to convert the value from radians to degrees. This results in a total of 30 multiplications and 14 divisions, which will take approximately 44 ms to calculate.

Pre-Calculated Lookup Table (Without Interpolation)

To determine an efficient implementation of a non-interpolated lookup table, the minimum number of entries required for guaranteed accuracy within 1° must first be determined. The most that a calculation can be off by is 1°. Therefore, if the result of some arctangent calculation is n°, the table must contain a greater result that is ≤ n + 1°, *or* a lesser result that is > n – 1°. The worst case would be that n is exactly 1° less than the greater result, *and* exactly 1° greater than the lesser result. In order to ensure that this worst case falls within the acceptable error of 1°, any two consecutive outputs of the lookup table must be ≤ 2° apart. Since the goal is to use the fewest number of table entries possible, 2° apart is the

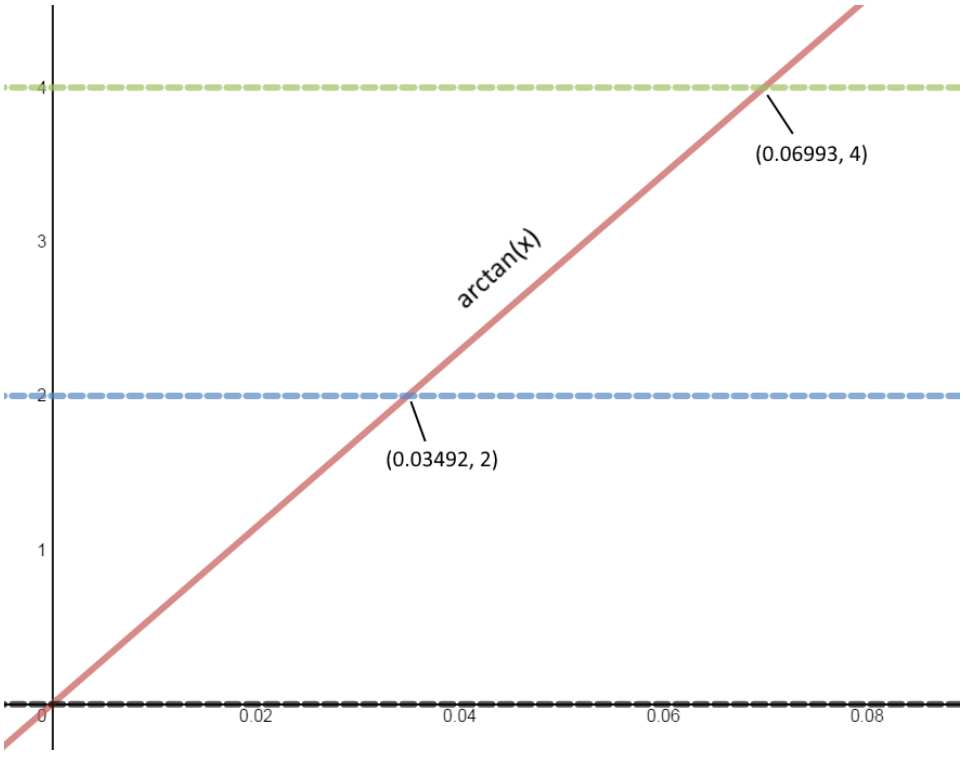

Figure 2. Graph of arctan(x) divided into 2° intervals (only first two are visible).

ideal difference between any two consecutive outputs of the lookup table. An easy way to find what entries this ideal table consists of is to divide a graph of arctangent into 2° intervals, as shown in Figure 2.

Each interval must have a single table entry that calculates to within 1° of the arctangent of all values within the interval. For example, the first interval results in values from 0° to 2°. Thus, the table entry to satisfy this interval should have an output of 1°, and should be referenced for any input value [0, 0.03492). The same process is used to calculate the first three entries of the lookup table, shown in Table 1 below.

TABLE 1
FIRST THREE ENTRIES OF THE LOOKUP TABLE

Entry #	Lower Input Bound	Upper Input Bound	Output (°)
1	0.00000	0.03492	1
2	0.03492	0.06993	3
3	0.06993	0.10510	5

Notice that the upper bound of any given entry is the same as the lower bound for the next entry. Because of this redundancy, in an actual software implementation only one of the bound numbers as well as the output need to be stored in the table. The table in its entirety can be seen in Appendix A.

A lookup table requires 30 words of overhead in memory, plus an additional word of memory for each entry to the table. The entire table consists of 23 entries. Thus, a non-interpolated, pre-calculated lookup table implementation requires 53 words of memory. Furthermore, because all calculation was done beforehand, SCOMP does not need to do any calculations other than finding the correct entry in the table. Therefore, this implementation of arctangent will run in an insignificant amount of time.

Pre-calculated Lookup Table (With Interpolation)

When linearly interpolating, an unknown input must be compared to both a lesser and greater known input. Therefore, both bounds of the domain, zero and one, must be entries into the table. Starting from the low end of the table, it becomes a question of how far away from zero the second entry can be while ensuring that the interpolated line (see Figure 3) is always within 1° of the actual graph. Because

the graph of arctan(x) is concave down (increasing at a decreasing rate), any linear interpolation between two known points will be an underestimate.

Thus, a visual way to find a good second entry to the table is to graph arctan(x) and arctan(x) – 1 on the same plane. Then, by trial and error, one can find a line that passes through (0, 0) and some second point, while never intersecting with arctan(x) – 1. The x value of this second point is an ideal second entry into the graph – it is far away enough from zero to minimize the number of entries into the table, but is still close enough to ensure that the error is no more than 1°. This process was carried out, and 0.525 was found to be a good input value for the second entry in the table (see Figure 4). As it turns out, a straight line can be drawn through (0.525, 27.700) and (1, 45) as well, while still staying within 1° of error. Thus, only three entries are needed for a pre-computed lookup table with interpolation to calculate arctangent to within the required accuracy. All of the entries are shown in Table 2.

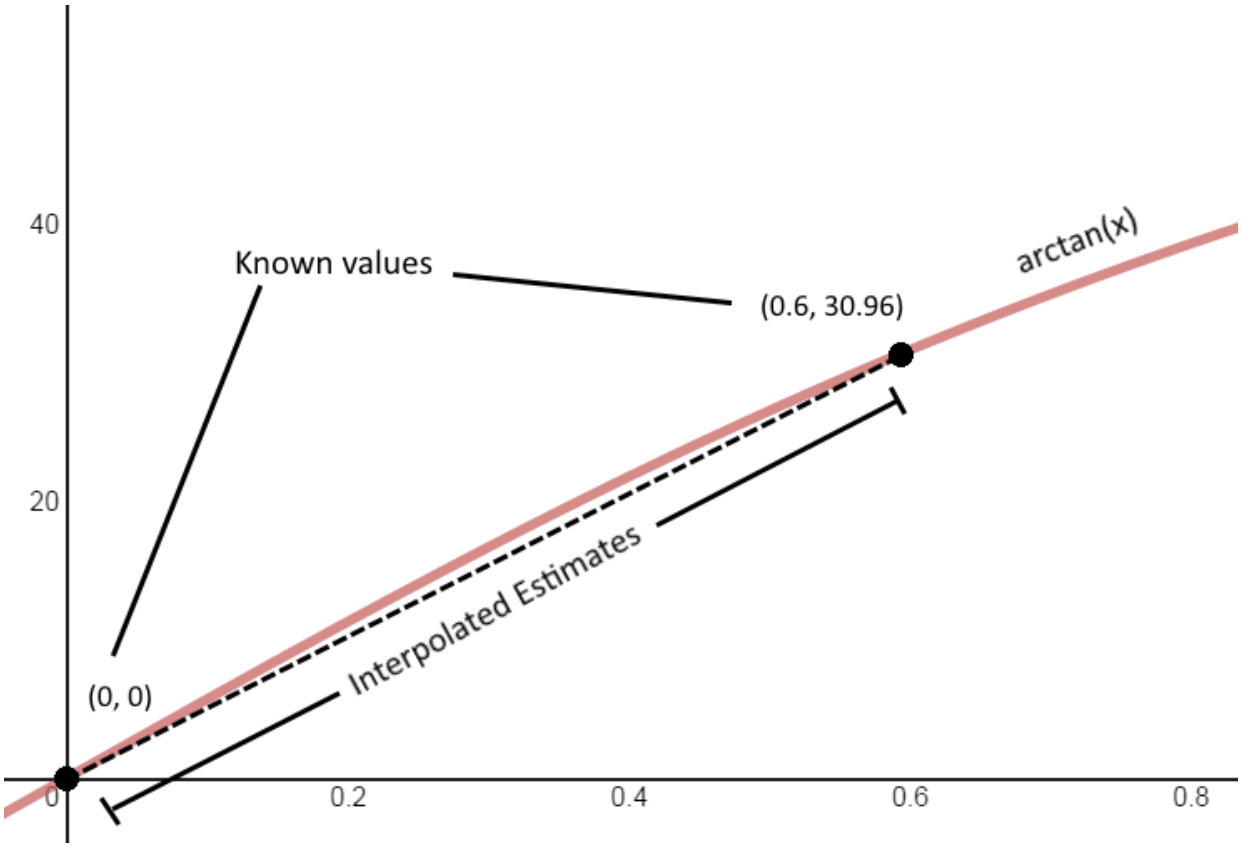

Figure 3. A graph of arctan(x), with two arbitrary values chosen to be "known values." Any value between these two must be calculated by interpolation. The interpolated result will always be an underestimate, due to the concavity of the graph.

TABLE 2

PRE-CALCULATED LOOKUP TABLE (WITH INTERPOLATION)

Entry #	Input	Output (°)
1	0	0
2	0.525	27.700
3	1	45

A lookup table with interpolation requires 60 words of overhead in memory, plus an additional word of memory for each entry to the table. A interpolated lookup table with three entries will have a memory footprint of 63 words.

To calculate an arctangent using this table, SCOMP must perform a few calculations. First, it looks up the table entry (*L*) whose input is immediately lower than the value to be calculated, as well as the entry (*H*) whose input is immediately higher than the value to be calculated. This is a simple table lookup, and happens in an insignificant amount of time. Then, it performs the following calculation to interpolate between the two values:

$$\arctan(x) \approx L_{output} + \frac{H_{output} - L_{output}}{H_{input} - L_{input}} \cdot (x - L_{input})$$

This calculation requires one division, and one multiplication. Each of these two operations requires 1ms, giving this implementation a calculation time of 2ms.

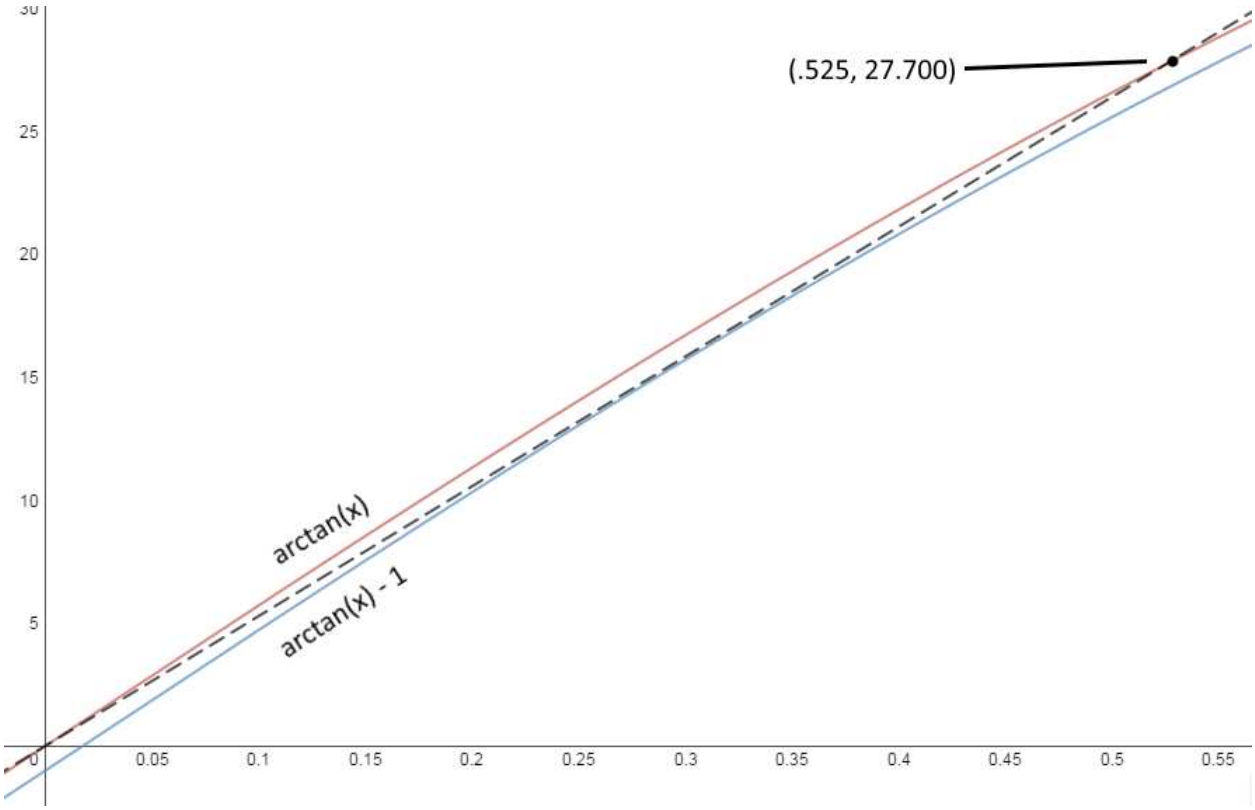

Figure 4. All interpolated values (dotted line) between (0, 0) and (0.525, 27.700), fall within the 1° margin of error.

Recommendation

It is recommended that a pre-calculated lookup table without interpolation be used to implement the calculation of arctangent for the DE2Bot. First and foremost, it has the smallest memory footprint of the three proposed solutions. It takes up only 53 words of the SCOMP's available memory, compared to 63 words for an interpolated lookup table implementation, and 200 words for a Taylor series implementation. This is the biggest advantage of using a non-interpolated lookup table, as having a small memory footprint is a larger concern than having a fast calculation time (as the calculation only needs to be performed once, at the start of the DE2Bot's journey).

As an added bonus, since the processor need not do any multiplications or divisions (the slowest operation for the SCOMP), the execution time will be the fastest of the three. An interpolated lookup

table will take approximately 2ms to calculate a result, while a Taylor series will take approximately 44ms.

An interpolated lookup table is comparable to a non-interpolated one in terms of both memory and time, but a non-interpolated table has a slight edge in both categories, as well as being the simpler of the two to implement. For these reasons, the recommended implementation for calculating an arctangent in software is using a non-interpolated lookup table.

Appendix A: Pre-Calculated Lookup Table Without Interpolation

Entry #	Lower Input Bound	Upper Input Bound	Output (°)
1	0.00000	0.03492	1
2	0.03492	0.06993	3
3	0.06993	0.10510	5
4	0.10510	0.14054	7
5	0.14054	0.17633	9
6	0.17633	0.21256	11
7	0.21256	0.24933	13
8	0.24933	0.28675	15
9	0.28675	0.32492	17
10	0.32492	0.36397	19
11	0.36397	0.40403	21
12	0.40403	0.44523	23
13	0.44523	0.48773	25
14	0.48773	0.53171	27
15	0.53171	0.57735	29
16	0.57735	0.62487	31
17	0.62487	0.67451	33
18	0.67451	0.72654	35
19	0.72654	0.78129	37
20	0.78129	0.83910	39
21	0.83910	0.90040	41
22	0.90040	0.96569	43
23	0.96569	1.03553	45