Design and Implementation of Variable Precision Algorithm for Transcendental Functions

Hao Jiangwei, Xu Jinchen, Guo Shaozhong, Xia Yuanyuan, Liu Dan
State Key Laboratory of Mathematical Engineering and Advanced Computing, Information Engineering University, Zhengzhou, China
1903377641@qq.com

Abstract—Mathematical function is an essential part of numerical program, and it is also the key factor that affects the precision and performance of a program. In the design of mathematical functions, in order to satisfy most application scenarios, it is necessary to correctly round and cover the calculated interval as much as possible. But in certain applications, correct rounding and full coverage of definition domain may not be required. Therefore, mathematical functions can be customized according to application requirements to avoid precision waste as well as improve performance. However, manually implementing mathematical functions is a time-consuming and error-prone task. Tools like Metalibm is designed to automatically generate mathematical function code, which is difficult to take advantage of all the mathematical properties of a function and results in the generated code performing slowly than the corresponding mathematical library function. A method of generating variable precision code based on mathematical properties is proposed for the transcendental function in this work. Experiments showed the performance of the proposed method is comparable to Glibc mathematical function.

1. INTRODUCTION AND MOTIVATION
The mathematical library (libm) is an important part of the basic software library, which provides support for mathematical operations and is a key factor affecting high performance computing. In the design of mathematical functions, in order to cover a wide range of application scenarios, all intervals in the defined domain are usually covered as far as possible, and the accuracy limit of the current floating point format is approximated as well [1]. However, certain applications may not require a large calculation interval or a high calculation accuracy. Too large calculation interval and too high precision will cause performance waste. For example, in earthquake engineering [2], trigonometric functions are used to calculate the seismic waves, which only need a limited interval \([-\pi, \pi]\). In addition, in the field of artificial intelligence, the calculation result can generally meet the requirements with three decimal places after the decimal point. According to the specific requirements of the application, manual customization of key mathematical functions is supposed to be a way to improve performance. However, considering the large number of interrelated parameters in the mathematical function, it is inefficient to manually implement the mathematical functions one by one. Therefore, this paper proposed an automatic design scheme of variable precision algorithm for the transcendental function.
2. BACKGROUND
The transcendental function cannot get exact result through finite addition, subtraction, multiplication, division, power and square root operations, and can only get approximate results in the computer [3]. There are many algorithms for transcendental functions approximation, such as table lookup methods, polynomial approximation methods, etc.

2.1 Analysis of existing methods
1) Table Lookup Method
Ideally, the table lookup method is to store the calculation results of all possible input values by precomputed table, and look up while execution. For the function \( y = f(x) \), assuming that the length of the input value is \( n \) bits, then \( n \times 2^n \) bits are needed to store the data. Taking double-precision floating-point numbers as an example, the length is 64 bit, and the required memory size is \( 64 \times 2^{34} \text{bit} = 2^{32} \text{TB} \), which is obviously unrealistic. Therefore, generally in the actual algorithm design process, the function calculation interval is sampled according to a certain strategy, and the accurate calculation result of the sampling point is stored in the table, and the approximate value of the function is obtained by looking up the table during the calculation.

2) Polynomial Approach Method
According to mathematical theory [4], any continuous function can be approximated infinitely by polynomials.

Theorem: (Weierstrass) Let \( f(x) \) be a continuous function on \([a, b]\). For any given \( \epsilon < 0 \), there exists a polynomial \( p(x) \), such that

\[
\max_{x \in [a, b]} |f(x) - p(x)| < \epsilon
\]

Polynomial approximation is an important method for transcendental functions. Literature [5] and [6] described this method in detail.

The polynomial approximation method is generally used for function approximation in small intervals for the function \( f(x), x \in [a, b] \), find a polynomial \( p(x) (p(x) \in P_n) \) to optimally minimize the "distance" \( \|p^* - f\| \).

The "distance" is the error in the general sense. There are basically two ways to evaluate the distance:

For the least square approximation, which minimizes the "average error", the "distance" can be expressed as

\[
\|p^* - f\|_{2, [a, b]} = \sqrt{\int_a^b w(x)(p^*(x) - f(x))^2 \, dx}
\]

Where \( w(x) \) is a continuous non-negative weight function, and the accuracy of the partial interval is picked higher by selecting the partial interval of \([a, b]\).

For the minimax approximations, which minimizes the worst-case error, the "distance" can be expressed as

\[
\|p^* - f\|_{\infty, [a, b]} = \max_{x \in [a, b]} \|p(x) - f(x)\|
\]

When \( w(x) = 1 \), the "distance" represents an absolute error. When \( w(x) = \frac{1}{f(x)} \), the "distance" indicates the relative error.

In the case of no ambiguity, in order to simplify the expression, \( \|p^* - f\|_{2, [a, b]} \) is used instead of \( \|p^* - f\|_{2, [a, b]} \), and \( \|p^* - f\|_{\infty, [a, b]} \) is used instead of \( \|p^* - f\|_{\infty, [a, b]} \).

There are many kinds of approximate polynomials. The Taylor polynomial and the minimax error polynomial are introduced below.
Taylor polynomial

Taylor polynomial form is simple. Taylor polynomial has the best convergence around the expansion point, and the approximation error increases with distance from the expansion point. The source of error for the Taylor polynomial is mainly the remainder including G.peano's remainder and Lagrange remainder. Taking \( \sin(x) \) as an example, the G.peano's remainder is \( o(x^2) \) in the following formula.

\[
\sin(x) = x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + o(x^7)
\]

Minimax error (fpminmax) polynomial

Take \( w(x) = 1 \), there are \( \| f - p^* \|_e = \max_{x \in [a,b]} \| f(x) - p(x) \| \). The target is \( \| f - p^* \|_e = \min_{p \in \mathbb{P}_n} \| f - p \|_e \). Such a polynomial \( p^* \) exists and is unique. It is called the degree-\( n \) minimax approximation polynomial to \( f(x) \) on \( [a,b] \). Chebyshev studied it and proposed the following theorem.

Theorem: (Chebyshev) \( p^* \) is the minimax degree-\( n \) approximation to \( f \) on \( [a,b] \), if and only if there exist at least \( n+2 \) values

\[
a \leq x_0 \leq x_1 \leq x_2 \leq \cdots \leq x_{n+1} \leq b,
\]

Such that

\[
p^*(x_i) - f(x_i) = (-1)^i[p^*(x_0) - f(x_0)] = \pm \| f - p \|_e.
\]

According to the above theorem proposed by Chebyshev, Remez designed the algorithms, which can find the minimax degree-\( n \) approximation polynomial of continuous function through iterative calculation.

2.2 Analysis of advantages and disadvantages of existing methods

1) Table Lookup Method

The table lookup method is computationally fast, because only the offset of the input value relative to the starting point needs to be calculated, and the corresponding function value can be directly returned or the approximate value of the returned result after fitting. However, the accuracy of the table lookup method is not good enough, especially when the interval is large, its memory consumption increases exponentially. To reduce the memory footprint, one should increase the distance between the interpolation points, which may sacrifice accuracy. In a small interval, a balance between accuracy and memory consumption can be achieved. But as the interval expands, it would be difficult to reach a balance.

2) Polynomial approximation

The polynomial approximation method is beneficial due to its light dependence on memory space and high precision. A polynomial is evaluated by a series of fused multiply-add (FMA) instructions in computer. The speed of the FMA instructions have a great influence on the performance of the polynomial evaluation.

2.3 Related work

Metalibm is a code generator for mathematical function developed by Christoph Lauter and Olga Kupriianova [7], which can generate mathematical function codes of any range and precision. The input to Metalibm can be a mathematical expression, target accuracy, implementation domain, etc.

First, Metalibm look up the mathematical properties of the function through black box tests, such as periodicity and symmetry, which facilitates implementation. Then, the computation domain is split into smaller subdomains, so that the function can complete the polynomial approximation at a reasonable cost in each small interval [8]. Finally, integrate these subdomains to generate function code.

Metalibm can be used for the automatic generation of many mathematical functions, but detecting mathematical properties is difficult by its method. Moreover, as the domain expands, the number of
subdomains that need to be split increases, resulting in an increase in memory space and a slower speed. Therefore, Metalibm is more suitable for approximation calculations in small intervals.

3. VARIABLE PRECISION FUNCTION ALGORITHM DESIGN

The above-mentioned table lookup method and polynomial approximation methods are applicable to function approximation calculations in small intervals in general, and each has its own applicable function range. These two algorithms can control the calculation accuracy by adjusting the algorithm parameters, but they often cause memory space occupation and performance changes. It is difficult to balance accuracy and performance for these two methods, and they do not make full use of the mathematical properties of the functions. Therefore, in the actual transcendental function algorithm design process, instead of using a single algorithm, but combining the advantages of each algorithm, making full use of the mathematical characteristics of each function, the design can meet the target accuracy and performance as high as possible.

For \( f(x) = \sin(x) \) on the domain \([0, 10\pi]\), with 52 correct bits.

At the first, using the periodicity and symmetry of the trigonometric function, the calculation interval can be transformed to \([0, \pi]\). If the polynomial approximation for the \([0, \pi]\) is used, the degree-13 Taylor polynomial with a coefficient precision of 54 bits can satisfy the target precision requirement.

Further, considering the advantage of the table lookup method, the \([0, \pi]\) interval is uniformly interpolated, and the number of interpolation is \(2^7\). Auxiliary entries are constructed using the exact results of the interpolation points. Polynomial approximation is used between the two interpolation points, thereby improving the computational accuracy and performance of the function.

At this time, the degree-6 Taylor polynomial with the coefficient precision of floating-point number of 54 effective digits in binary can satisfy the target precision requirement. Finally, using the transformation formula of \(\sin(x)\), the construction table and the polynomial approximation are combined to obtain the calculated value of \(\sin(x)\).

As can be seen from the above examples, in the process of the constructor algorithm, several parameters appear, such as polynomial times, polynomial coefficient precision, number of table interpolation points, and so on. By adjusting these parameters, you can control the calculation accuracy, space occupancy, and performance of the function. The main part of the function in the above example can be regarded as a function template. These parameters are called function template parameters.

3.1 Design Idea of Function Algorithm Implementation with Controllable Accuracy

When designing a generic function, it is often necessary to cover the function definition domain and maintain a high precision. The design of the custom function only needs to meet the target precision in the specified interval, and its performance potential is greater than the general function.

If the implementation domain and target accuracy are not high, even the table lookup method can be used to complete the function design. In addition, in the function design process, you also need to consider the characteristics of the target architecture, such as whether to support FMA operations. If not supported, the proportion of polynomial approximation in the algorithm should not be too large.

In general, it is difficult to obtain the calculation accuracy of the function through static analysis in the algorithm design stage, and the large-scale data tests are used to obtain the calculation accuracy of the function. Therefore, it is impossible to obtain a function that satisfies the accuracy requirement in one time design, so it is necessary to optimize the function template parameters continuously and iteratively through testing until an efficient and space-saving function algorithm that satisfies the target precision is realized.

Based on this design idea of this algorithm, the process of designing the variable precision function algorithm is as follows:
Step 1: Enter the target function and its implementation interval, desired accuracy, and some hardware constraints, such as whether to support FMA operations, cache size, and so on. According to the input parameters and hardware constraints, determine the function implementation method, or say, the function template; and then select the appropriate function template parameters to form the parameter space.

Step 2: Search for the best parameter vector in the parameter space, and continuously optimize the function algorithm based on the function template. At the beginning of the search, the parameters are initialized to the minimum values. At the time,

\[
\text{input parameters \\& hardware constraints} \rightarrow \text{function template \\& parameters space} \\
\rightarrow \text{generate} \rightarrow \text{initial parameter vector} \\
\begin{cases}
\text{if satisfy?} \\
\text{no} \rightarrow \text{search for better parameter vector} \\
\text{yes} \rightarrow \text{the final code}
\end{cases}
\]

Figure 1. design process of the variable precision function
the precision is the smallest and the space is the smallest. According to the initial parameter vector, generate function code and then test the calculation accuracy. If the desired accuracy requirement is not met, then change the parameter values step by step to improve the accuracy until the desired accuracy requirement is met. Then, based on the parameter vector, search for the nearby parameter vector that satisfies the accuracy requirement until the performance is significantly degraded.

Step 3: According to the function evaluation standard, among the parameter vectors satisfying the accuracy requirement, pick the optimal parameter vector, and generate the final function code.

The design flow is shown in the Figure 1.

3.2 Variable precision algorithm implementation for \(\sin(x)\)
For analyzing the characteristics and implementation process of the variable precision function algorithm design and implementation better, \(\sin(x)\) is taken as an example.

First, according to the bound of the implementation domain \(I = [a, b]\)
, the offset
\[
X = \left\lfloor \frac{a + b}{2} \right\rfloor \frac{2\pi}{2} \quad \text{is calculated. A new interval} \quad [a_i, b_i] \quad \text{is obtained, where} \quad a_i = \frac{b - a}{2} + \frac{a + b}{2} - X,
\]
\[
b_i = \frac{b - a}{2} + \frac{a + b}{2} - X \quad \text{. By shifting the calculation interval near to the zero point, the cumulative error is reduced so that the calculation accuracy is effectively guaranteed. At the time,} \quad x' = x - X \quad \text{, there are}
\]
\[
\sin(x') = \sin(x - X) = \sin(x), x' \in [a_i, b_i] \quad \text{. Using the mathematical properties of the trigonometric function, the interval can be transformed to} \quad [0, \frac{\pi}{2}] \quad \text{. At the time, there are}
\]
\[
x'' = x' - k \frac{\pi}{2}, x' \in [0, \frac{\pi}{2}], k = \left\lfloor \frac{x'}{\frac{\pi}{2}} \right\rfloor \quad \text{. In}
this process, it is necessary to subtract an integral multiple of \( \frac{\pi}{2} \) from \( x' \). Considering that \( \pi \) cannot be accurately represented in computer, floating point numbers with high precision are used to represent \( \pi \) generally for minimizing the error caused by \( \pi \). Here, a parameter, namely the effective number of bits of \( \pi \), is involved. The larger \( s_1 \) is, the higher the precision of \( \pi \) is, and the smaller the error caused by the transform interval is.

Then, according to the mathematical properties of \( \sin(x) \), let \( x'' = \frac{1}{2^p} \times \frac{\pi}{2} \times x' \), \( \Delta x = x'' - x''' \), there is \( \sin(x'') = \sin(x'' + \Delta x) = \sin(x''') \cos(\Delta x) + \cos(x''') \sin(\Delta x) \), \( \Delta x \in [0, \frac{\pi}{2^p}] \). The values of \( \sin(x''') \) and \( \cos(x''') \) can be calculated in advance and stored in the table as coefficients. Here two parameters are used, the table item density \( m \) and the table item precision \( 2^s \). The larger \( m \) is, the smaller the interval of \( \Delta x \) is, and the better the convergence of the polynomial approximation is. The larger \( 2^s \) is, the higher the coefficient accuracy and the smaller the error.

As for \( \sin(x''') \) and \( \cos(x''') \), an approximate polynomial expansion is sufficient. The approximate polynomial \( p(x) \) uses three parameters, the polynomial order \( n \), the polynomial coefficient precision \( 3^s \), and the polynomial type \( k \). The larger \( n \) and \( 3^s \) are, the smaller the polynomial approximation error is. \( k \) is selected according to different function types and hardware constraints information.

\( s_1, s_2, s_3, m, n \) and \( k \) together form the parameters space of the \( \sin \) function template. The accuracy and performance of the function code generated according to each parameter vector are successively tested, and finally the code that satisfies the accuracy requirement and has the best performance is selected and provided to the user.

4. PERFORMANCE TEST

This article uses a lateral comparison approach to test the performance of custom functions implemented using automated design solutions in the paper. The performance test compare the mathematical functions in the Glibc standard library and those generated by Metalibm.

The performance test method used in the paper is to calculate the number of beats required to run the function. After a lot of experimental analysis, the performance curve of the algorithm tends to be stable when the function runs more than 100,000 times [9].

Therefore, several different size intervals in the \( \sin(x) \) definition domain are selected, each of which contains 100,000 random floating-point numbers. Within these intervals, performance tests were performed on custom functions implemented using automated design schemes in the paper, Glibc math functions. Then use the deviation method to find the average value of 100,000 data, which is the number of beats in a single execution of the function.

Through the lateral comparison test analysis, with the Glibc mathematics, function performance can be compared. As the Figure 2 shows.
Figure 2 performance test for $\sin(x)$

5. CONCLUSIONS AND FUTURE WORK
This paper first introduces the table lookup method and polynomial approximation method, deeply analyzes their advantages and disadvantages, and discusses the feasibility of combining two kinds of algorithm design to implement the variable precision transcendental function. An automatic design scheme of variable precision transcendental function algorithm based on iteration, modularization and parameterization is proposed for transcendental function. Lateral tests show that the performance of the function code generated by the automatic design scheme in this paper is about the same as that of Glibc function, which verifies the superiority of this scheme. In the future, the automatic design scheme will be used to improve the algorithms of other transcendental functions in the libm, and realize an automatic generation system of libm.

ACKNOWLEDGMENT
Thanks should go to all the teachers and friends who helped me a lot on the draft.
Supported Project: the National Natural Science Foundation of China (No.61802434)

REFERENCES
[1] Xu JC, Guo SZ, Han YZ, Wang L, and Zhou B, "Exception Handling Approach of Floating Mathematical Functions," Journal of Software, 2015.
[2] Ranf, Richard Tyler, "Damage to bridges during the 2001 Nisqually earthquake," In Proceedings of the 2001 Earthquake Engineering Symposium for Young Researchers. Ed. by Andrea Dargush, Phillip Gould, and Gerard Pardoen. Multidisciplinary Center for Earthquake Engineering Research, pp. 273-293. 2001.
[3] Han SX, “Algorithm research and hardware implementation of floating point trigonometric transcendental functions,” Northwestern polytechnical university, 2006.
[4] Rudin, Walter, “Principles of mathematical analysis,” Principles of mathematical analysis / McGraw-Hill, 1976.
[5] Jean-Michel Muller, "Elementary functions: algorithms and implementation," Mathematics & Computer Education 34.1(1997):21–52.
[6] Jean-Michel Muller, Nicolas Brisebarre, Florent de Dinechin, Claude-Pierre Jeannerod, Vincent Lefèvre, Guillaume Melquiond, et al., HANDBOOK OF FLOATING-POINT ARITHMETIC. Handbook of floating-point arithmetic. Birkhäuser, 2010.
[7] N. Bruniè, F. D. Dinechin, O. Kupriianova, and C. Lauter, "Code generators for mathematical functions," IEEE Symposium on Computer Arithmetic IEEE Computer Society, 2015.
[8] Kupriianova Olga, and Christoph Lauter, "A domain splitting algorithm for the mathematical functions code generator," In 2014 48th Asilomar Conference on Signals, Systems and Computers, pp. 1271-1275. IEEE, 2014.
[9] Guo SZ, Xu JC, Chen JX, “Improved Transcendental Function General Algorithm,” Computer engineering, 2012, 38(15):31-34.