ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

УДК 004.021
DOI: 10.18101/2304-5728-2019-4-57-64

БИБЛИОТЕКА МАТЕМАТИЧЕСКИХ ФУНКЦИЙ ДЛЯ ЯЗЫКА ФУНКЦИОНАЛЬНО-ПОТОКОВОГО ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ ПИФАГОР [1]

© Удалова Юлия Васильевна
кандидат технических наук, доцент,
Сибирский федеральный университет
Россия, 660041, г. Красноярск, пр-т Свободный, 79
E-mail: judalova@sfu-kras.ru

Язык функционально-потокового параллельного программирования Пифагор является оригинальным языком программирования, его ключевые особенности — это отсутствие переменных и операторов цикла, поддержка параллелизма на уровне операций, выполнение операций по готовности данных.
Инструментальные средства языка Пифагор развиваются, разрабатывается репозиторий функций. Реализована библиотека математических функций, по функциональности сравнимая с библиотекой math.h языка C. Вычисление функций базируется на рядах Маклорена и формулах приведения. Часть математических функций реализована в двух версиях, одна из которых выполняет быстрые и менее точные вычисления, а другая предполагает точные и более медленные вычисления.
Реализация библиотеки математических функций для языка Пифагор выполнена впервые, вычисление математических функций поддерживает возможности распараллеливания на уровне операций. Математические функции включены в открытый репозиторий функций языка Пифагор, тогда как во многих языках программирования математические библиотеки предполагают только функциональные вызовы, не открывая разработчику те математические методы, алгоритмы и программный код, с помощью которых они производят вычисления.
Ключевые слова: функциональное программирование; потоковое программирование; параллельное программирование; архитектурно-независимое программирование; параллелизм на уровне операций; алгоритмы математических функций; приближенные вычисления; ряды Маклорена; репозиторий функций; информационно-управляющий граф.

Для цитирования:
Удалова Ю. В. Библиотека математических функций для языка функционально-потокового параллельного программирования Пифагор // Вестник Бурятского государственного университета. Математика, информатика. 2019. № 4. С. 57–64.

[1] Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований, грант № 17-07-00288.

Введение

В настоящее время язык функционально-потокового параллельного программирования Пифагор [1] и его инструментальные средства [2–6] активно развиваются. Статья посвящена реализации библиотеки математических функций для языка Пифагор и иллюстрирует особенности потоковых параллельных вычислений с параллелизмом на уровне операций.

Значительная часть математических функций вычисляется с помощью рядов Маклорена. Используется как более быстрая и менее точная организация вычислений, при которой вычисляется предопределенное количество элементов ряда с заранее просчитанными коэффициентами, так и менее быстрая и более точная организация вычислений, при которой элементы ряда вычисляются динамически до достижения нужной точности.

Приближенное вычисление математических функций и их программная реализация в ряде известных языков программирования к настоящему моменту являются задачами решенными. Для языка Пифагор и его оригинальной модели вычислений задача решена впервые. Кроме этого, реализация математических функций во многих известных языках скрыта от разработчика, он может вызвать выполнение функции, но не может знать, с помощью какого метода и алгоритма происходит вычисление результата. Таким образом, описанные в статье основные принципы проектирования математической библиотеки могут быть полезны тем читателям, которым требуется решить подобную задачу для собственной вычислительной системы.

1 Быстрое приближенное вычисление математических функций на примере синуса, арксинуса и экспоненты

Ряд Маклорена для синуса имеет следующий вид.

$$\sin(x) \approx x - x^3/3! + x^5/5! - x^7/7! + ..., \quad для\ |x| \le 1. \qquad (1)$$

При вычислении синуса для произвольного угла = x радиан требуется привести аргумент в отрезок $[-\pi, \pi]$, используя свойство периодичности функции. Если аргумент принадлежит $[-\pi, -\pi/2) \cup (\pi/2, \pi]$, он передвигается в отрезок $[-\pi/2, \pi/2]$, при этом запоминается флаг последующего умножения результата на -1. Для аргумента x из $[-\pi/2, \pi/2]$ вычисляется $x' = 2x/\pi$, формула (1) принимает следующий вид.

$$\sin(\pi x'/2) \approx (\pi/2)x' - (\pi/2)^3 x'^3/3! + (\pi/2)^5 x'^5/5! - ..., для\ |x'| \le 1. \quad (2)$$

Для быстрого вычисления приемлемого результата берутся первые шесть слагаемых ряда и заранее вычисляются вне программного кода их численные коэффициенты, это:

$$\pi/2 = 1.570796326794,\ -(\pi/2)^3/3! = -0.645964097505,$$
$$(\pi/2)^5/5! = 0.079692626245,\ -(\pi/2)^7/7! = -0.0046817541,$$
$$(\pi/2)^9/9! = 0.00016044118,\ -(\pi/2)^{11}/11! = -0.0000035988432.$$

В процессе работы функции вычисляются степени аргумента, производится умножение на коэффициенты и суммирование искомой формулы. Такой алгоритм сравним с аналогичным последовательным императивным алгоритмом, за исключением того, что модель вычислений языка Пифагор предполагает потенциальное параллельное выполнение всех тех операций, относящихся к одной или разным функциям, входные данные которых известны. Реализация обозначенного алгоритма средствами языка Пифагор формирует следующие последовательности операторов, потенциально доступные для параллельного вычисления (рис. 1).

Рис. 1. Упрощенный информационно-управляющий граф функции sin

Рисунок 1 иллюстрирует то, что множество операторов одной ветви графа, например, вычисление степеней аргумента x2, x3, x5, x7 и т. д. могут быть выполнены только последовательно, т. к. нуждаются в результатах вычислений меньших степеней, тогда как операторы одного слоя, например «(x2,x5):*>>x7», «(0.079692626245,x5):*>>p3», «(p1,p2):+>>S1», потенциально могут быть доступны для параллельного выполнения.

При иллюстрации графа на рисунке 1 повсеместно использовано выделение вспомогательных идентификаторов, это x2, … x11, p1, … p6, S1, … S5. Программный код функции может быть реализован в таком же стиле,

либо с помощью объединения формул в одно выражение. Например, вычисление искомой суммы в коде описано как «(((((p1,p2):+,p3):+,p4):+,p5):+,p6):+ >> return», при этом порядок вычисления операторов остается таким же, как на графе (рис. 1). Но, например, выделение в коде идентификатора x2 является желательным, его наличие позволяет не производить операцию x'*x' несколько раз при вычислении степеней аргумента.

Сравнение результатов быстрого приближенного вычисления синуса в языке Пифагор и вычисления синуса с помощью вызова функции sin из библиотеки math.h в языке C представлено в таблице 1.

Таблица 1

Вычисление синуса в языке Пифагор, в языке C (библиотека math.h) и в онлайн-калькуляторе

Аргумент x	math.h	Пифагор	Калькулятор
0.2	0.198668	0.198669	0.198669
1.2	0.932039	0.932039	0.932039
3.2	–0.058373	–0.058374	–0.058374
10.0	–0.544021	–0.544021	–0.544021

Другие тригонометрические функции могут быть вычислены аналогично через их ряд Маклорена либо с помощью подстановки значений синуса в формулы приведения. При реализации библиотеки математических функций для языка Пифагор использовались формулы приведения.

При приближенном вычислении некоторых функций с помощью ряда Маклорена может наблюдаться значительное снижение точности в окрестностях отдельных точек, лежащих в области определения ряда, как правило, это 1 и –1. К таким функциям относится арксинус. Для арксинуса области определения функции и ряда совпадают, это [–1, 1], поэтому преобразование аргумента, подобное тому, которое было выполнено для синуса, не требуется. При приближенном вычислении арксинуса используется следующая формула (3), основанная на его ряде Маклорена, с вычисленными вне программного кода коэффициентами.

$$\begin{aligned}\arcsin(x) \approx x &+ 0.166666666666x^3 + 0.075x^5 + 0.044642857142x^7 + \\ &0.030381944444x^9 + 0.022372159090x^{11} + 0.017352764423x^{13} + \\ &0.01396484375x^{15} + 0.011551800896x^{17} + 0.009666219208x^{19} + \\ &0.008390335809x^{21} + 0.007312525873x^{23} + 0.000011679728x^{25}.\end{aligned} \quad (3)$$

Если для достижения приемлемой точности синуса достаточно взять первые шесть слагаемых, то для арксинуса желательно вычислить не менее двенадцати слагаемых. Дополнительно, если x расположен в окрестностях точек 1 или –1, точность приближенной формулы значительно снижается. При реализации арксинуса для языка Пифагор при x, принадлежащем $[-1, 0.93) \cup (0.93, 1]$, используется вспомогательная формула

$$\arcsin(x) = \pi/2 - \arcsin\left(\sqrt{1-x^2}\right).$$

При вычислении экспоненты используется следующая формула с предварительно вычисленными коэффициентами, основанная на ряде Маклорена.

$$e^x \approx 1 + x + 0.5x^2 + 0.166666666666\text{x}^3 + 0.041666666666\text{x}^4 + 0.008333333333\text{x}^5 + 0.001388888888\text{x}^6 + 0.00019841269841\text{x}^7, \ для |x| \le 0.7. \quad (4)$$

Ряд Маклорена экспоненты определен на [–1, 1], но так как при быстром приближенном вычислении используется конечное малое число элементов, для достижения приемлемой точности выбрана область [–0.7, 0.7]. Область определения экспоненты — это вся числовая ось, поэтому для произвольного аргумента x используется свойство экспоненты $e^{a+b} = e^a e^b$. Аргумент x разделяется на целую и дробную части, e в дробной степени рассчитывается с помощью формулы 4, e в целой степени вычисляется перемножением констант e или 1/e, для повышения скорости вычислений при этом применяются дополнительные вычисленные вне программного кода величины, например $e^8 = 2980.957987041726$, $e^{64} = 6.235149080811582\text{e+}027$.

2 Точное рекурсивное вычисление математических функций на примере логарифма

Ряд Маклорена для логарифма натурального имеет следующий вид.

$$\ln(x+1) \approx x - x^2/2 + x^3/3 - x^4/4 + \ldots, \ для \ -1 < x \le 1. \quad (5)$$

Формула (5) включает большие коэффициенты, чем ряды Маклорена рассмотренных выше функций, поэтому быстрое вычисление логарифма натурального не реализовано, так как оно потребовало бы перечисления значительного количества, до нескольких сотен, возможно и тысяч, слагаемых. Вместо этого используется рекурсивное вычисление формулы (5) до достижения выбранной точности 10^{-8}, в императивном языке это можно было бы реализовать с помощью цикла, в языке Пифагор циклические алгоритмы реализуются с помощью рекурсивных функций.

Область определения логарифма натурального — это множество положительных чисел, поэтому для произвольного аргумента x используется свойство логарифма натурального

$$\ln(b10^a) = \ln(b) + a\ln(10) = \ln(b) + a\,2.302585092994046,$$

где b можно выбрать так, чтобы оно подходило для вычисления ln(b) с помощью формулы (5).

Рекурсивная функция ожидает аргумент вида (число1, число2, число3), где число1 это b, число2 при первом вызове рекурсии совпадает с b, число3 — это знаменатель ряда, при первом вызове рекурсии равно единице. Одна итерация рекурсии вычисляет четыре слагаемых ряда Маклорена. Упрощенный информационно-управляющий граф функции логарифма натурального имеет следующий вид (рис. 2).

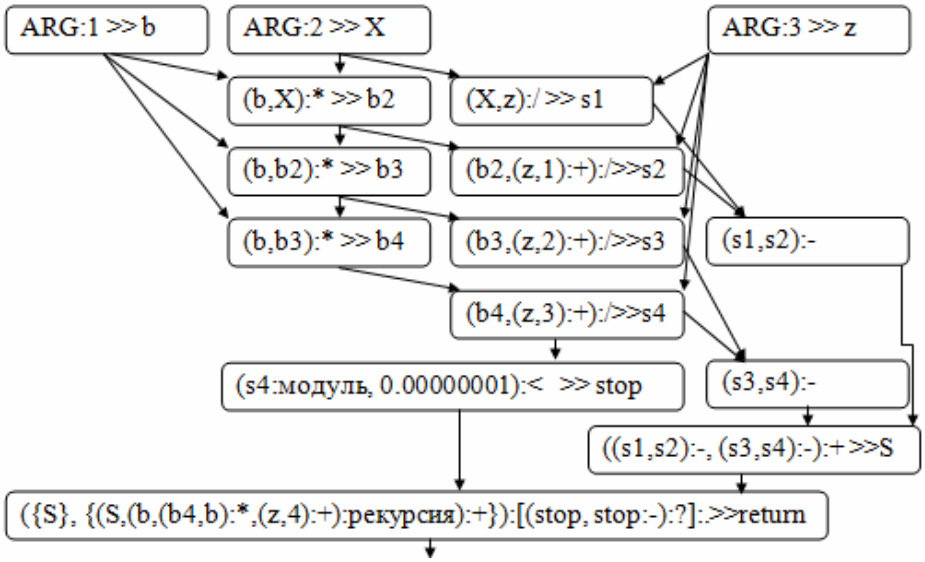

Рис. 2. Упрощенный информационно-управляющий граф функции ln

Выражения, заключенные в фигурные скобки, например {(S, (b, (b4,b):*,(z,4):+):рекурсия):+} на рисунке 2, относятся к задержанным вычислениям и будут выполняться только при истинности условия stop:-, т. е. когда четвертое слагаемое s4 в текущей итерации, взятое по модулю, оказалось больше, чем 10^{-8}. Команды (b, (b4,b):*,(z,4):+) формируют аргумент вида (число1, число2, число3) для следующей итерации этой же рекурсивной функции. Представленное здесь рекурсивное вычисление формулы (5) можно реализовать средствами языка Пифагор и с помощью других способов организации рекурсии, но при реализации библиотеки математических функций выбран приведенный способ.

Таблица 2

Вычисление логарифма натурального в языке Пифагор, в языке С и в онлайн-калькуляторе

Аргумент x	math.h	Пифагор	Калькулятор
0.07	–2.659260	–2.659260	–2.659260
1.25	0.223144	0.223144	0.223144
200.0	5.298317	5.298317	5.298317
10500.0	9.259130	9.259131	9.259131
100000000.0	18.420681	18.420679	18.420680

Имея реализацию вычисления логарифма натурального, можно вычислить логарифм по произвольному основанию, используя свойство $\log_a(b) = \ln(b)/\ln(a)$, для десятичного логарифма в знаменатель подставляется вычисленная вне кода константа 2.302585092994046. Возведение в степень реализовано с помощью формулы $x^y = e^{y\ln(x)}$.

Заключение

Задача создания математической библиотеки для языка Пифагор полностью реализована и является одной из подзадач, успешно выполненных при исполнении гранта РФФИ «Архитектурно-независимая разработка параллельных программ на основе функционально-потоковой парадигмы».

Перечисление реализованных математических функций. Модуль. Синус. Косинус. Тангенс. Котангенс. Арксинус. Арккосинус. Арктангенс. Арккотангенс. Секанс. Косеканс. Корень квадратный. Корень кубический. Округление. Округление вниз. Округление вверх. Целая и дробная части числа. Экспонента. Гиперболические синус, косинус, тангенс, котангенс. Логарифм натуральный. Логарифм десятичный. Логарифм по указанному основанию. Возведение в степень. Целая часть от деления и остаток от деления (аргументы функции могут быть числами, как целыми, так и вещественными). Выделение мантиссы и показателя степени двойки. Умножение числа на двойку в степени. Знак числа. Гиперболические ареакосинус, ареасинус, ареатангенс, ареакотангенс, ареасеканс, ареакосеканс. Возведение двойки в степень. Максимум из пары чисел. Минимум из пары чисел. Гипотенуза. Модуль разности.

Все реализованные функции включены в открытый репозиторий языка Пифагор, таким образом, разработчик может не только исполнять и просматривать их код, но и заимствовать его для создания собственных алгоритмов или альтернативной реализации математических функций.

Литература

1. Легалов А. И., Ушакова М. С. Особенности разработки и преобразования функционально-потоковых параллельных программ // Суперкомпьютерные дни в России: тр. междунар. конф. М., 2018. С. 999–1000.

2. A toolkit for the development of data-driven functional parallel programmes / А. И. Легалов [и др.] // Communications in Computer and Information Science. 2018. Т. 910. С. 16–30. DOI: 10.1007/978-3-319-99673-8_2.

3. Ушакова М. С., Легалов А. И. Верификация программ со взаимной рекурсией на языке Пифагор // Моделирование и анализ информационных систем. 2018. Т. 25, № 4 (76). С. 358–381. DOI: 10.18255/1818-1015-2018-4-358-381.

4. Васильев В. С., Легалов А. И. Оптимизация инварианта цикла в языке Пифагор // Моделирование и анализ информационных систем. 2018. Т. 25, № 4 (76). С. 347–357. DOI: 10.18255/1818-1015-2018-4-347-357.

5. Инструментальная поддержка создания и трансформации функционально-потоковых параллельных программ / А. И. Легалов [и др.] // Труды Института системного программирования РАН. 2017. Т. 29, № 5. С. 165–184. DOI: 10.15514/ISPRAS-2017-29(5)-10.

6. Удалова Ю. В., Легалов А. И. Верификация функционально-потоковых параллельных программ методом индуктивных утверждений // Доклады Академии наук высшей школы Российской Федерации. 2014. № 2–3 (23–24). С. 125–132.

LIBRARY OF MATHEMATICAL FUNCTIONS FOR THE LANGUAGE OF DATA-DRIVEN FUNCTIONAL PARALLEL PROGRAMS PYTHAGOR

Yulia V. Udalova
Cand. Sci. (Engineering), A/Prof.,
Siberian Federal University
79 Svobodny Prospect, Krasnoyarsk 660041, Russia
E-mail: judalova@sfu-kras.ru

The language of data-driven functional parallel programs Pythagor is an original programming language, which key features are the absence of variables and loop operators, support for operation-level parallelism, operating by availability.

The Pythagor language tools are developed, and a repository of functions is worked out. A library of mathematical functions in terms of functionality comparable to the library math.h of C language has been created. Function calculation is based on the Maclaurin series and reduction formulas. Some mathematical functions are implemented in two versions, one of which performs fast and less accurate calculations, and the other involves accurate and slower calculations.

The implementation of the library of mathematical functions for the Pythagor language has been performed for the first time; the calculation of mathematical functions supports parallelization capabilities at the level of operations. Mathematical functions are included in the open repository of the Pythagor language.

Keywords: functional programming; streaming programming; parallel programming; architecture-independent programming; operation-level parallelism; algorithms of mathematical functions; approximate calculations; Maclaurin series; function repository; data control flow graph.

References

1. Legalov A. I., Ushakova M. S. Osobennosti razrabotki i preobrazovaniya funktsionalno-potokovykh parallelnykh programm [Features of the Development and Transformation of Data-Driven Functional Parallel Programs]. *Supercomputernie dni v Rossii — Supercomputer Days in Russia.* Proc. Int. Conf. Moscow. 2018. Pp. 999–1000.

2. Legalov A. I. et al. A Toolkit for the Development of Data-Driven Functional Parallel Programmes. *Communications in Computer and Information Science*. 2018. No. 910. Pp. 16–30. DOI: 10.1007/978-3-319-99673-8_2.

3. Ushakova M. S., Legalov A. I. Verifikatsiya programm so vzaimnoi rekursiei na yazyke Pifagor [Verification of Programs with Mutual Recursion in Pythagor]. *Automatic Control and Computer Sciences*. 2018. Vol. 25, No. 4(76). Pp. 358–381. DOI: 10.18255/1818-1015-2018-4-358-381.

4. Vasilyev V. S., Legalov A. I. *Optimizatsiya invarianta tsikla v yazyke Pifagor* [Optimization of Loop Invariant in the Pythagor Language]. 2018. Vol. 25, No. 4(76). Pp. 347–357. DOI: 10.18255/1818-1015-2018-4-347-357.

5. Legalov A. I. et al. Instrumentalnaya podderzhka sozdaniya i transformatsii funktsionalno-potokovykh parallelnykh programm [Tool Support for Creation and Transformation of Data-Driven Functional Parallel Programs]. *Programming and Computer Software*. 2017. Vol. 29, No. 5. Pp. 165–184. DOI: 10.15514/ISPRAS-2017-29(5)-10.

6. Udalova Yu. V., Legalov A. I. Verifikatsiya funktsionalno-potokovykh parallelnykh programm metodom induktivnykh utverzhdenii [Verification of Data-Driven Functional Parallel Programs by Inductive Assertions]. *Proceedings of the Russian Higher School Academy of Sciences*. 2014. No. 2–3 (23–24). Pp. 125–132.