Таблично ориентированный подход к нахождению значений функций одной вещественной переменной

В.Г. Абрамов, Н.В. Баева, А.А. Казаков, С.Ю. Соловьев

***Аннотация*—В статье рассматриваются ключевые аспекты технологии программной реализации таблично ориентированных функций вещественной переменной. Применительно к функциям в описании технологии термин "вычисление" означает выработку значения функции для заданного аргумента посредством некоторого численного метода, а термин "нахождение значения" подразумевает, что для выработки значения функции применяется поиск в таблице по (возможно, модифицированному) аргументу. Главная проблема технологии состоит в построении таблицы значений заданной функции для всех аргументов из некоторого стандартного диапазона. При этом погрешность вычислений и область определения функции определяются двоичным форматом представления чисел, а переход от значений функции для аргументов из стандартного диапазона к значениям функции для всей области определения не влияет на окончательную точность нахождения значений.**

В статье представлены и обоснованы требования, которым должны удовлетворять методы формирования таблиц значений: во-первых, методы должны учитывать особенности и особенности процесса формирования, во-вторых, методы должны обеспечивать верификацию вычисленных значений. В интересах реализации и массового применения таблично ориентированных функций вычисленные значения предлагается эффективно сжимать с тем, чтобы обеспечить их относительно компактное хранение и быстрый поиск.

Для исследования технологии предлагается использовать набор модельных функций, включающий элементарные и специальные функции. Основные этапы технологии проиллюстрированы на примере логарифмической функции. В заключении приводятся обнадеживающие результаты практической проверки технологии на тестовых функциях.

***Ключевые слова*—функция, алгоритм, упаковка, поиск.**

I. Введение

Первые вычислительные машины создавались в интересах численных расчетов, которые всегда используют элементарные и/или специальные функции *sin(·)*, *cos(·)*, *Γ(·)* и др. При этом по необходимости полагалось, что методы вычисления таких функций [1] должны быть предельно компактны, в частности, коэффициенты и значения функций "не должны загромождать память машины" [2]. В связи с этим для каждой новой ЭВМ приходилось разрабатывать собственные методы вычисления стандартных функций. Так, для ЭВМ Урал-1 [3] с 36-ти разрядной памятью для вычисления *tg(x)* применялась [4] формула

$$tg(x) \approx x\frac{a_0+(a_1+a_2x^2)x^2}{a_0+[a_3+(a_4+a_5x^2)x^2]x^2}, \quad (1)$$

где $a_0 = +0.5$, $a_1 = -0.060606060$,

$a_2 = +0.001010101$, $a_3 = -0.227272727$,

$a_4 = +0.010101010$, $a_4 = -0.000048100$.

Формула (1) применима для $0 \le x < \pi/4$, она обеспечивает погрешность вычисления $1/2^{36}$.

Параллельно с прогрессом вычислительной техники методы вычисления функций развились в самостоятельный и нетривиальный раздел численного анализа [5], [6]. Между тем времена изменились. Выросли и продолжают расти объемы доступной оперативной памяти. Кроме того, появились мощные вычислительные комплексы, способные сгенерировать и разместить в памяти все без исключения значения часто используемых функций для популярных форматов чисел. При этом на ординарных компьютерах нахождение значения функции сводится к поиску в специально организованной таблице значений.

В настоящей работе будем различать термины “нахождение значения” и “вычисление”. Первый термин означает поиск значения функции для заданного аргумента в известной таблице значений. Второй – означает вычисление значения функции посредством того или иного численного метода. Не ограничивая общности, можно полагать, что таблицы значений заполняются посредством вычислений. Будем использовать обозначение $Tab(f;x)$ как эквивалент фразы “найденное в таблице значение функции *f* для аргумента *x*”. В общем случае $Tab(f;x) \approx f(x)$.

Научная задача предлагаемого исследования состоит в разработке новых быстрых методов нахождения значений, не стесненных (в разумных пределах) объемами оперативной памяти для хранения коэффициентов и/или таблиц значений.

Статья получена 10 декабря 2016.
Абрамов В.Г., МГУ имени М.В. Ломоносова (email: vlabr@cs.msu.su)
Баева Н.В., МГУ имени М.В. Ломоносова (email: nbaeva@gmail.com)
Казаков А.А., МГУ имени М.В. Ломоносова (email: mail@akazakov.ru)
Соловьев С.Ю., МГУ имени М.В. Ломоносова (email: soloviev@glossary.ru)
Исследование выполнено за счет гранта Российского фонда фундаментальных исследований (проект No. 16-07-00858).

II. Структурный анализ научной задачи

Естественная декомпозиция исходной научной задачи предполагает решение следующих шести подзадач.

1. Выбор и исследование класса модельных функций.
2. Выбор алгоритмов вычисления табличных значений модельных функций (см. п.III).
3. Выбор стандартных диапазонов и генерации значений модельных функций (см. п.IV).
4. Разработка способов хранения таблиц модельных функций (см. п.V).
5. Разработка таблично ориентированных алгоритмов нахождения значений модельных функций (см. п.VI).
6. Исследование полученных алгоритмов.

Набор конкретных модельных функций, задействованных для отработки технологии, должен быть в достаточной мере представительным, исследованным и разнообразным. Другими словами, набор модельных функций должен

- состоять из общеизвестных и популярных функций;
- минимизировать издержки на разработку алгоритмов вычисления модельных функций; и
- демонстрировать все особенности технологии.

Учитывая сказанное, в модельный набор функций включены тригонометрические, показательные и степенные функции, а также функции им обратные. Кроме того, в модельный набор в качестве представителя класса специальных функций включена гамма-функция.

По мере приближения к решению главной – шестой подзадачи – возможны пересмотры ранее полученных результатов. Исследование построенных алгоритмов состоит в получении сравнительных характеристик вновь построенных и традиционных алгоритмов вычисления модельных функций, а также в составлении и обосновании прогноза востребованности полученных алгоритмов. Исходными данными для прогноза являются мировые тенденции в наращивании вычислительных мощностей и оценки объемов таблиц модельных функций. Приведем постановки и рассмотрим особенности основных подзадач.

III. Выбор алгоритмов вычисления табличных значений модельных функций

В соответствии с программистской традицией под вещественным числом понимается выражение $\sigma 2^{\rho}\cdot\alpha$, в котором

σ – знак числа, $\sigma\in\{+,-\}$;

ρ – двоичный порядок числа, ρ – целое;

α – мантисса числа, $0.5\le\alpha<1$.

Каждое вещественное число, за исключением нуля, представляется единственной тройкой $<\sigma,\rho,\alpha>$. Например, $-2.5=-2^1\cdot 0.625$. Двоичное представление мантиссы имеет вид $0.1\delta_1\delta_2...$, где $\delta_i\in\{0,1\}$. В этом представлении последовательность нулей и единиц $\delta_1\delta_2...$ есть изменяемая часть мантиссы. Допустимый спектр значений вещественных чисел определяется его форматом, который, помимо прочего, фиксирует максимальные количества двоичных разрядов для размещения порядка ρ и изменяемой части мантиссы. В большинстве современных вычислительных машин используются форматы одинарной и двойной точности стандарта IEEE 754 [7], соответствующие типам *float* и *double* языка C++. Изменяемые части мантисс типов *float* и *double* содержат соответственно 23 и 52 разрядов. Таким образом, для этих типов существуют 2^{23} и 2^{52} различных мантисс.

По определению, алгоритм вычисления табличных значений модельной функции для

(a) модельной функции *f*,

(b) вещественного *x* и

(c) длиной изменяемой части мантиссы *m*

вычисляет вещественное число *z* такое, что

$$|f(x)-z|<\varepsilon,\ \text{где}\ \varepsilon=1/2^{m+1}.$$

Если $0.5\le f(x)<1$, то искомое число $z=0.1\delta_1...\delta_m$ получается округлением числа $0.1\delta_1...\delta_m\delta_{m+1}$, двоичное представление которого совпадает с $f(x)$ вплоть до *m*+2-го разряда в дробной части.

Классические методы вычисления функций – суть методы вычисления табличных значений модельных функций – хорошо известны [5]. Кроме того, существуют и иные методы, позволяющие вычислять заданное количество разрядов в двоичном представлении вещественных чисел. К таким методам, в частности, относятся рекурсивные методы, а также методы вытеснения.

Рекурсивные методы исходят из специальных представлений функций, а также из их асимптотических оценок в предельных точках. Типичные [8], [9] рекурсивные представления выглядят так:

$$\sin(x)=2\sin(\frac{x}{2})\cdot[1-2\sin^2(\frac{x}{4})],$$

$$\ln(1+x)=\ln(1+\frac{x}{x+2})-\ln(1-\frac{x}{x+2})\cdot$$

Причем

- $\sin(x)\sim x$ и $\ln(1+x)\sim x$ при $x\sim 0$.
- $|x/(x+2)|<|x|$ при $-1<x$ и $x\ne 0$.

Заметим, что формулу

$$1+x=\frac{1+x/(x+2)}{1-x/(x+2)}\qquad (для\quad x>0)$$

можно также применять для вычисления степенной функции.

Методы вытеснения [10] по сути дела собирают из нулей и единиц искомые значения функций. Рассмотрим в качестве примера вычисление методом вытеснения функции $\log_2(x)$. Во-первых, для вычисления любого двоичного логарифма достаточно уметь вычислять $\log_2(\cdot)$ для аргументов α_0 из $[0.5,1)$. Во-вторых, зафиксируем $K=m+2$ и обозначим $p_0, \ldots, p_K$, монотонно возрастающую последовательность вещественных чисел, в которой $p_n=2^{-1/2^n}$:

$$p_0=0.5,\ p_1\approx 0.707,\ p_2\approx 0.841,\ p_3\approx 0.917,\ \ldots;$$
$$p_n\to 1\ \text{при}\ n\to\infty.$$

Вычисление $\log_2(\alpha_0)$ состоит в вычислении монотонно неубывающей последовательности чисел $\alpha_0, \alpha_1, \ldots, \alpha_K$ по следующему правилу:

$$\alpha_{n+1} = \begin{cases} \alpha_n / p_{n+1}, & \text{если} \quad \alpha_n \in [p_n, p_{n+1}), \\ \alpha_n & \text{иначе}. \end{cases} \qquad (2)$$

Параллельно с вычислением чисел α_n формируется искомое значение логарифма (с заданной точностью)

$$\log_2(a_0) \approx -\sum_{n=1}^{K} \frac{\delta_n}{2^n}, \text{ где } \delta_n = \begin{cases} 1, & \text{если} \quad \alpha_n \neq \alpha_{n-1}, \\ 0 & \text{иначе}. \end{cases}$$

Заметим, что описанный метод зависит от значений констант $p_0, \ldots, p_K$, которые можно вычислить заранее. Более того, если метод дополнить константами $q_1 = 1/p_1, \ldots, q_K = 1/p_K$, то в формуле (2) выражение α_n / p_{n+1} можно заменить на $\alpha_n \cdot q_{n+1}$.

IV. Выбор стандартных диапазонов и генерация значений модельных функций

Какими бы блестящими ни были перспективы роста объемов оперативной памяти, вычислять значения для всевозможных аргументов не имеет смысла; достаточно иметь таблицы значений для некоторого стандартного диапазона аргументов. При этом каждая модельная функция характеризуется

- стандартным диапазоном значений;
- операцией разложения аргумента; и
- операцией сборки искомого значения.

Стандартный диапазон представляет собой подмножество из области допустимых значений модельной функции. Пусть x – произвольный аргумент из области допустимых значений. Применительно к задаче нахождения числа $z \approx f(x)$:

- операция разложения приводит исходную задачу для аргумента x к нахождению одного или нескольких значений для аргументов из стандартного диапазона.
- операция сборки конструирует число z из значений функций для аргументов, принадлежащих стандартному диапазону.

Задача нахождения стандартного диапазона известна также как задача уменьшения аргумента или задача приведения аргумента [11]. Для функции $\log_2(x)$

- стандартный диапазон значений есть $[0.5,1)$;
- операция разложения выделяет в заданном аргументе x порядок и мантиссу: $x = 2^{\rho}\alpha$;
- операция сборки: $z = \rho + Tab(\log_2; \alpha)$.

Для функции $\ln(x)$ стандартный диапазон и операция разложения не изменяются, а операция сборки выглядит так: $z = Tab(\ln; \alpha) - \rho \cdot Tab(\ln; 0.5)$.

Важно отметить, что операции разложения и сборки, как правило, используют арифметические операции над вещественными числами, а, значит, вносят в искомый результат погрешность, которую следует оценить.

В простейшем случае формирование таблиц модельных функций сводится[1] к перебору всех аргументов из стандартного диапазона и к вычислению соответствующих элементов таблицы. При этом многократно применяется некоторый подходящий численный метод. Разумеется, при таком подходе автоматически предполагается использование современных многопроцессорных вычислительных комплексов [12]. Результатом работы алгоритма формирования является линейный список значений функций, который имеет смысл лишь в связи с фиксированным порядком перечисления аргументов.

Для логарифмической функции со стандартным диапазоном $[0.5,1)$ и форматом одинарной точности типа *float* в таблице необходимо разместить “всего-навсего” 2^{23} чисел общим объемом $4*2^{23} = 32$ Mb.

В некоторых случаях учет особенностей задачи формирования позволяет существенно сократить трудоемкость построения списка значений. Если, например, при формировании списка для функции $\log_2(x)$ аргументы из стандартного диапазона перечисляются в порядке убывания, то при вычислении логарифма очередного аргумента $\alpha \in [p_n, p_{n+1})$ можно воспользоваться соотношением

$$\log_2(\alpha) = \log_2(\alpha / p_{n+1}) + \log_2(p_{n+1}),$$

а поскольку $\alpha < \alpha / p_{n+1}$, то $\log_2(\alpha / p_{n+1})$ находится в ранее построенной части списка:

$$\log_2(\alpha) \approx Tab(\log_2; \alpha / p_{n+1}) - 1/2^{n+1},$$

или $\log_2(\alpha) \approx Tab(\log_2; \alpha q_{n+1}) + (-1/2^{n+1})$.

Таким образом, при вычислении очередного значения логарифма помимо логических команд используются лишь две арифметические команды: одно умножение и одно сложение.

Особую задачу подготовки списка значений составляет верификация полученных значений. Понятно, что полученный список обязан отвечать общим свойствам модельной функции. Если, скажем, модельная функция является монотонно возрастающей, то список значений должен быть неубывающей последовательностью[2]. Кроме того, особенности отдельных функций порождают специфические способы проверки значений. Для проверки логарифмических функций, например, можно использовать разложения аргументов на два сомножителя. Как показало численное исследование типа *float*, диапазон $[0.5,1)$ содержит $2^{23} = 8388608$ различных чисел-аргументов, 4792227 из которых (57%) раскладываются в произведение чисел из того же диапазона:

$$\frac{8388609}{2^{24}} = \frac{12582912}{2^{24}} \cdot \frac{11184812}{2^{24}} \quad \text{и т.д.}$$

Для каждого такого равенства $a = b \cdot c$ с учетом погрешности вычислений для логарифмической функции должны выполняться соотношения

$$|Tab(\log; b) + Tab(\log; c) - Tab(\log; a)| < 3\varepsilon.$$

V. Разработка способов хранения таблиц модельных функций

Преобразование линейного списка значений в таблицу, пригодную для практического использования, предполагает сжатие информации без потерь. При этом к алгоритму сжатия предъявляются противоречивые

[1] В рамках выбранного формата представления чисел.

[2] Если значения рассматриваются в порядке возрастания аргументов.

требования. С одной стороны таблица должна быть компактной, с другой – таблица должна обеспечивать эффективный выбор значений. Компромиссное решение достигается посредством учета конкретных особенностей той или иной модельной функции.

Общий подход к построению таблицы предполагает разбиение линейного списка значений на блоки, каждый из которых сжимается отдельно. Доступ к блокам обеспечивает относительно компактная индекс-структура. При построении блоков различают два подхода: (1) все блоки имеют равную длину, (2) длины блоков могут изменяться в определенных пределах. Второй подход ухудшает теоретические оценки быстродействия, однако с его помощью удается построить компактные таблицы для сильно варьирующих модельных функций.

Известные в настоящее время [13] методы сжатия табличных функций можно условно разделить на два основных класса:

- алгоритмы сжатия общего назначения: LZMA, LZMA2, PPM и др.;
- алгоритмы сжатия с предварительным преобразованием данных: дельта-кодирование, линейно-предсказывающее кодирование и др.

Алгоритмы сжатия общего назначения для декодирования конкретного числа требуют декодирования всех его предшественников в блоке, что существенно сказывается на быстродействии алгоритмов нахождения значений.

Потенциально, алгоритмы с предварительным преобразованием данных обеспечивают более высокую степень сжатия, однако их применение затрудняется необходимостью выбирать конкретные значения большого количества управляющих параметров.

VI. Разработка алгоритмов нахождения значений модельных функций

С содержательной точки зрения предложенный подход к нахождению значений функций вещественной переменной описывается объектом, в состав которого входят:

- стандартный диапазон и собственно таблица значений;
- методы, реализующие операции разложения и сборки;
- метод нахождения значения функции для заданного аргумента.

В настоящее время таблично ориентированные методы нахождения значений функций вещественной переменной представляют собой один из многих вариантов будущего развития информатики. В связи с этим указать точный метод реализации искомых функций не представляется возможным, однако можно привести некоторые, в определенном, смысле полярные концепции такой реализации.

Первая концепция, исходящая из приоритетного роста объемов оперативной памяти, состоит в размещении реализующего объекта целиком в памяти локального компьютера. Вторая концепция, исходящая из опережающего развития сетевых технологий, состоит в размещении таблицы значений на удаленном сервере, а доступ к элементам таблицы обеспечивается соответствующими сетевыми запросами.

VII. Заключение

Работоспособность описанного подхода проверена [14] на четырех модельных функциях *float* → *float*, использующих библиотеку math.h языка C++. Принятые в эксперименте стандартные диапазоны не накладывают ограничений на аргументы функций, то есть объем линейного списка значений составляет 4 Gb. В результате проведенных испытаний установлено, что для выбранных модельных функций наилучшим вариантом упаковки является метод равных блоков, предварительно подвергнутых дельта-кодированию. При этом объем сжатых таблиц (для алгоритма LZMA2) составляет от 20 до 70 Mb, а соответствующие функции с использованием процессора Intel Core-i7 3770k позволяют находить от 8 до 11 миллионов значений в секунду. Столь обнадеживающие оценки для типа *float* ставят на повестку дня проверку описанного подхода для типа *double*. Вместе с тем, серьезное увеличение количества разрядов в мантиссе неизбежно актуализирует теоретические исследования в области численного анализа.

Библиография

[1] Люстерник Л.А., Абрамов А.А., Шестаков В.И., Шура-Бура М.Р. Решение математических задач на автоматических цифровых машинах. – М.: Изд-во АН СССР, 1952.

[2] Березин И.С., Жидков Н.П. Методы вычислений, том 1. – М.: ГИФМЛ, 1959.

[3] Смирнов Г.С. Семейство ЭВМ “Урал-1”. Страницы истории разработок. – Пенза, 2005.

[4] Люстерник Л.А., Червоненкис О.А., Янпольский А.Р. Математический анализ. Вычисление элементарных функций. – М.: Физматгиз, 1963.

[5] Попов Б.А., Теслер Г.С. Вычисление функций на ЭВМ. – Киев: Наукова думка, 1984.

[6] R. Brent, P. Zimmermann, Modern Computer Arithmetic. Cambridge University Press, 2011.

[7] Яшкардин В.Л. IEEE 754: стандарт двоичной арифметики с плавающей точкой. [Электронный ресурс]. Страница доступа: http://www.softelectro.ru/ieee754.html.

[8] Стефанюк В.Л. Рекурсивное оценивание арифметических функций в системах ЛИСП. Программирование, 1981, No.5. С. 92-94.

[9] Сальников М.С. Рекурсивный алгоритм вычисления логарифма. // Информационные процессы, том 12, No. 3, 2012. С. 248-252

[10] Соловьев С.Ю. Алгоритм вычисления логарифмов методом вытеснения. // Вестн. Моск. ун-та сер. 15 Вычисл. матем. и киберн., 2013, No. 2. С. 38-43

[11] Теслер Г.С. Адаптивные аппроксимации и итеративные процессы. // Математические машины и системы, том 2,. 2004. С.22-41

[12] Воеводин В.В., Воеводин Вл.,В Параллельные вычисления. – СПб.: БХВ-Петергург, 2002.

[13] Ватолин Д., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М.: ДИАЛОГ-МИФИ, 2003.

[14] Казаков А.А. Исследование алгоритмов упаковки таблично заданных функций // Сборник тезисов лучших выпускных работ факультета ВМК МГУ. – М.: МАКС Пресс, 2015. С. 93-95.

The table-driven approach to finding one real variable functions values

Vladimir Abramov, Natalia Baeva, Artem Kazakov, Sergey Solov'ev

Abstract—**In this article, we consider main aspects of software implementation of the technology table-driven real variable functions. With regard to functions, the term "computation" (in the description of the technology) means the fabrication function values (for a given argument) by some numerical method, and the term "finding value" means the fabrication function values by search in the table using the (possibly modified) argument as a key. The main problem of the technology is to construct a table of values of a given function to all the arguments of some of the standard range. Herewith the error estimation and function domain are determined by a binary format for numbers representation, and the transition from the function values for arguments from the standard range to the values from the function domain does not affect the error of finding the final value.**

We also present and justify the requirements to be met by the methods of formation of tables of values. To study the technology we propose to use a set of model functions. To demonstrate the basic stages of the technology we have chosen a logarithmic function. In conclusion, we present encouraging results of practical verification technology to test functions.

Keywords—**function, algorithm, compression, search.**