ПРОГРАММИРОВАНИЕ, 2022, № 6, с. 14–21

АНАЛИЗ ДАННЫХ

УДК 519.651

АЛГОРИТМ ВЫЧИСЛЕНИЯ КОРРЕКТНО ОКРУГЛЕННОГО ЗНАЧЕНИЯ ЭКСПОНЕНТЫ С ДВОЙНОЙ ТОЧНОСТЬЮ С ИСПОЛЬЗОВАНИЕМ АРИФМЕТИКИ РАСШИРЕННОЙ ДВОЙНОЙ ТОЧНОСТИ

© 2022 г. А. Н. Годунов[a,*] (ORCID: 0000-0001-5952-9185)

[a] *ФГУ Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук*

117218 Москва, Нахимовский просп., 36, корп. 1, Россия

*****E-mail: nkag@niisi.ras.ru*

Поступила в редакцию 01.06.2022 г.
После доработки 29.06.2022 г.
Принята к публикации 04.07.2022 г.

Использование функций, вычисляющих корректно округленное значение экспоненты позволяет, с одной стороны, получить наилучшее приближение, а с другой стороны, обеспечить переносимость (portability) программ. В статье предлагается алгоритм вычисления корректно округленного значения экспоненты, когда аргумент и значение функции являются числами двойной точности, а вычисления производятся с использованием чисел расширенной двойной точности. Дано формальное описание алгоритма. Представленный алгоритм реализован в виде функции на языке Си, проведено его тестирование и измерены временные характеристики. Проведено сравнение с другими алгоритмами.

DOI: 10.31857/S0132347422060036

1. ВВЕДЕНИЕ

Корректно округленное значение экспоненты есть результат округления математически точного значения экспоненты. Использование функций, вычисляющих корректно округленное значение экспоненты позволяет, с одной стороны, получить наилучшее приближение, а с другой стороны, обеспечить переносимость (portability) программ. Вычисление корректно округленного значения экспоненциальной функции рекомендовано стандартом IEEE 754 [1]. Если x – число двойной точности, не равное нулю, то e^x – трансцендентное число. В силу этого вычислить на компьютере точное значение экспоненты для ненулевого аргумента невозможно. Так как корректно округленное значение экспоненты – кусочно-постоянная функция, то можно ограничиться вычислением приближенного значения экспоненты. При этом погрешность вычисления экспоненты должна быть меньше, чем расстояние между точным значением экспоненты и ближайшей точкой разрыва функции округления.

Значения аргумента, для которых расстояние между точным значением экспоненты и ближайшей точкой разрыва функции округления минимально, считаются наиболее трудными (worst cases) для вычисления корректно округленного значения экспоненты. В. Лефевр, Ж.-М. Мюллер [2] были первыми, кто произвел поиск трудных случаев для вычисления корректно округленного значения экспоненциальной функции и других элементарных функций. В дальнейшем эта работа была продолжена В. Лефевром [3] и автором [4]. Результаты этих работ определили требования к точности вычисления экспоненты для получения корректно округленного значения этой функции.

В статье предлагается алгоритм вычисления корректно округленного значения экспоненциальной функции. Аргумент и значение функции – плавающие числа двойной точности, но при вычислениях используются плавающие числа расширенной двойной точности. Обычно для вычисления корректно округленного значения экспоненты с двойной точностью используется арифметика двойной точности. Единственным исключением, известным автору, является функция Лаутера (Lauter) [8], написанная на ассемблере и оптимизированная для процессора Intel Itanium. Эта функция использует числа расширенной двойной точности при вычислениях. Реализация предлагаемого алгоритма на языке Си (и поэтому мобильная) требует существенно меньше времени на вычисление корректно округленного значения экспоненты, чем функция Лаутера.

2. ПРЕДВАРИТЕЛЬНЫЕ СВЕДЕНИЯ

В этом разделе даны основные обозначения, определения и используемые результаты. Напомним, что для представления чисел двойной точности используется основание 2 и 53-битная мантисса. При вычислениях мы используем арифметику расширенной двойной точности: основание 2, длина мантиссы 64 бита.

Стандарт IEEE 754 [1] определяет следующие режимы округления: округление к ближайшему, округление к плюс бесконечности, округление к минус бесконечности и округление к нулю. Предусмотрено два вида режима округления к ближайшему: roundTiesToAway (к большему, если ближайших два) и roundTiesToEven (к числу с четной мантиссой, если ближайших два).

Предлагаемый алгоритм вычисляет корректно округленное значение e^x для всех режимов округления, но при вычислениях используется только округление к ближайшему (roundTiesToEven). Так как e^x всегда положительно, то округление к 0 эквивалентно округлению к минус бесконечности. В силу этого мы не будем рассматривать округление к 0. Если x – алгебраическое число не равное 0, то e^x – трансцендентное число. Поэтому при округлении e^x оба вида округления к ближайшему эквивалентны.

Определение 2.1. *Пусть X – арифметическое выражение, x – точное значение X, а x^* – вычисленное значение X. Величину $\Delta(x) = x^* - x$ будем называть абсолютной погрешностью вычисления.*

Погрешность вычисления обусловлена ошибками округления. Если x – действительное число, то через $\langle x\rangle$ будем обозначать округленное значение x в режиме округления к ближайшему (roundTiesToEven) при использовании арифметики плавающих чисел расширенной двойной точности.

Определение 2.2. *Пусть p и q – целые числа такие, что $p \ge q$. Множества $B^+_{p,q}$, $B^-_{p,q}$ и $B_{p,q}$ мы определяем следующим образом:*

- $B^+_{p,q}$ *– множество всех чисел вида*

$$x = b_p \cdot 2^p + b_{p-1} \cdot 2^{p-1} + \ldots + b_q \cdot 2^q,$$

где каждое из чисел $b_p, b_{p-1}, \ldots, b_q$ равно 0 *или* 1;

- $B^-_{p,q} = \{x | -x \in B^+_{p,q}\}$;

- $B_{p,q} = B^+_{p,q} \cup B^-_{p,q}$.

Определение 2.3. *Пусть n, m, p и q – целые числа такие, что $n \ge m > p > q$. Множество $T_{n,m,p,q}$ мы определяем следующим образом*:

$$T_{n,m,p,q} = \{(t_1, t_2, t_3) | t_1 \in B_{n,m}, t_2 \in B_{m-1,p},$$
$$|t_2| \le 2^{m-1}, t_3 \in B_{p-1,q}, |t_3| \le 2^{p-1}\}.$$

Определение 2.4. *Пусть n – целое, а x – действительное числа. Тогда $R_n(x)$ есть ближайшее к x число вида $m \cdot 2^n$, где m – целое. В случае неоднозначности выбираем четное m.*

Если n – целое, а x – действительное число, то

$$|R_n(x) - x| \le 2^{n-1}.$$

Если n – целое, а x – число расширенной двойной точности такие, что $|x| < 2^{n+62}$, и вычисление $x + 3 \cdot 2^{n+62}$ не приводит к переполнению, то

$$R_n(x) = \langle x + 3 \cdot 2^{n+62}\rangle - 3 \cdot 2^{n+62}.$$

Теорема 2.1 (алгоритм Fast2Sum [7]). *Пусть x и y – машинные числа такие, что $|x| \ge |y|$. Если их сложение не ведет к переполнению, то существуют такие машинные числа z и zz, что $z = \langle x + y\rangle$ и $x + y = z + zz$. Эти числа могут быть получены следующим образом:*

$$z := \langle x + y\rangle;$$
$$w := \langle z - x\rangle;$$
$$zz := \langle y - w\rangle.$$

Алгоритм Fast2Sum был предложен Деккером. Ниже мы будем использовать следующую нотацию для алгоритма Fast2Sum:

$$(z, zz) \leftarrow Fast2Sum(x, y).$$

Следующее определение введено В. Лефевром и Ж.-М. Мюллером (см. [2] и [7]).

Определение 2.5. *Пусть a и b – действительные числа, которые имеют одинаковый знак и не равны нулю, и q – целое, такие что $2^q \le |a|, |b| < 2^{q+1}$. Расстоянием между мантиссами a и b мы будем называть величину*

$$\frac{|a - b|}{2^q}.$$

Теорема 2.2 (В. Лефевр, Ж.-М. Мюллер [7]). *Пусть x – число двойной точности и y – его экспонента. Пусть, далее, y^* – приближенное значение y такое, что расстояние между мантиссами y и y^* ограничено ε. Тогда в следующих случаях*

- *если $|x| \ge 2^{-30}$ и $\varepsilon \le 2^{-113}$,*
- *если $2^{-54} \le |x| < 2^{-30}$ и $\varepsilon \le 2^{-158}$*

округление y^* эквивалентно округлению y для любого из рассматриваемых режимов округления.

Приведем две модификации теоремы 2.2. Первая относится ко всем рассматриваемым режимам округления.

Теорема 2.3. *Пусть x – число двойной точности и y – его экспонента. Пусть, далее, y^* – приближенное значение y, такое что расстояние между мантиссами y и y^* ограничено ε. Тогда в следующих случаях*

- $|x| \ge 2^{-37}$ *и* $\varepsilon \le 1.33 \cdot 2^{-113}$,
- $2^{-44} \le |x| < 2^{-37}$ *и* $\varepsilon \le 1.33 \cdot 2^{-134}$,
- $2^{-49} \le |x| < 2^{-44}$ *и* $\varepsilon \le 1.33 \cdot 2^{-149}$,
- $2^{-54} \le |x| < 2^{-49}$ *и* $\varepsilon \le 1.33 \cdot 2^{-158}$

округление y^* эквивалентно округлению y для любого из рассматриваемых режимов округления.

Доказательство следует из данных, приведенных в приложении А работы [4]. Вторая модификация относится к режимам округления к ближайшему (см. [4], теорема 6.1).

Теорема 2.4. *Пусть x – число двойной точности такое, что $|x| \ge 2^{-54}$, а y – его экспонента. Пусть, далее, y^* – приближенное значение y, такое что расстояние между мантиссами y и y^* ограничено $1.67 \cdot 2^{-112}$. Тогда для режимов округления к ближайшему округленное значение y^* равно округленному значению y.*

3. ОСОБЫЕ СЛУЧАИ

Положим

$$x_{ovr} = \texttt{0x1.62e42fefa39efp+9} \approx 709.78,$$
$$x_{dnrm} = -\texttt{0x1.6232bdd7abcd2p+9} \approx -708.40,$$
$$x_{zero1} = -\texttt{0x1.74385446d71c3p+9} \approx -744.44,$$
$$x_{zero2} = -\texttt{0x1.74910d52d3051p+9} \approx -745.13.$$

Пусть x – число двойной точности. С помощью непосредственных вычислений было получено следующее. Если $x > x_{ovr}$, то корректно округленное значение e^x равно $+\infty$ при всех видах округления, кроме округления к $-\infty$. При округлении к $-\infty$ оно равно $(2 - 2^{-52}) \cdot 2^{1023}$.

Если $x_{dnrm} \le x \le x_{ovr}$, то при всех видах округления корректно округленное значение e^x есть нормализованное число двойной точности.

Если $x < x_{dnrm}$, то при всех видах округления корректно округленное значение e^x есть ненормализованное число или 0.

Если $x < x_{zero1}$, то $e^x < 2^{-1074}$ (наименьшее положительное ненормализованное число). Поэтому корректно округленное значение e^x равно 0 в режиме округления к $-\infty$. В случае округления к $+\infty$ оно равно 2^{-1074}.

Если $x < x_{zero2}$, то $e^x < 2^{-1075}$. Поэтому в режиме округления к ближайшему и к $-\infty$ корректно округленное значение e^x равно 0. В случае округления к $+\infty$ оно равно 2^{-1074}.

4. АЛГОРИТМ

Предлагаемый алгоритм вычисления корректно округленного значения экспоненты состоит из нескольких (под)алгоритмов, которые реализуют стадии сокращения аргумента, аппроксимации и восстановления. Рассматриваемые алгоритмы используют при вычислениях арифметику расширенной двойной точности, а в результате их применения мы получаем корректно округленное значение экспоненты двойной точности.

4.1. Сокращение аргумента

Предполагается, что предварительно отфильтровываются нечисловые, а также слишком большие и слишком малые значения аргумента, которые обрабатываются отдельно. Детальное описание этого шага можно найти в [5]. Далее мы предполагаем, что аргумент принадлежит отрезку $[x_{zero2}, x_{ovr}]$.

Стадия сокращения аргумента состоит из четырех шагов (алгоритмы 4.1, 4.2, 4.3 и 4.4). Алгоритм 4.1 для сокращения аргумента использует равенство

$$e^x = 2^n \cdot e^{x - n\cdot \ln 2}.$$

Величину $\ln 2$ мы аппроксимируем тройкой $l = (l_1, l_2, l_3) \in T_{-1,-40,-77,-130}$. Путем непосредственных вычислений получаем

$$|l_1| < 0.694, \quad |l_2| < 1.517 \cdot 2^{-43},$$
$$|l_3| < 1.851 \cdot 2^{-79}, \qquad (4.1)$$
$$|\ln 2 - (l_1 + l_2 + l_3)| < 1.901 \cdot 2^{-137}.$$

Константу $1/\ln 2$ аппроксимируем числом расширенной двойной точности $\tilde{l}$:

$$\left|\frac{1}{\ln 2} - \tilde{l}\right| \le 2^{-64}. \qquad (4.2)$$

Алгоритм 4.1. Сокращение аргумента. Шаг 1

1: $m_0 \leftarrow R_0(x \cdot \tilde{l}\,)$
2: $x_{0,1} \leftarrow x - m_0 \cdot l_1 \triangleright x_{0,1}$ вычисляется точно, $|x_{0,1}| < 0.5 \cdot \ln 2 + 1.596 \cdot 2^{-33}$
3: $x_{0,2} \leftarrow -m_0 \cdot l_2 \quad \triangleright x_{0,2}$ вычисляется точно
4: $x_{0,3} \leftarrow -m_0 \cdot l_3 \quad \triangleright x_{0,3}$ вычисляется точно

Теорема 4.1. *Пусть x – плавающее число двойной точности такое, что $x_{zero2} \le x \le x_{ovr}$, числа m_0, $x_{0,1}$, $x_{0,2}$ и $x_{0,3}$ получены с помощью алгоритма* 4.1. *Тогда m_0 – целое число, $x_{0,1}$ и $x_{0,2}$ – плавающие числа двойной точности, а $x_{0,3}$ – плавающее число расширенной двойной точности, и справедливы следующие равенства и неравенства*:

$$x = m_0 \cdot \ln 2 + (x_{0,1} + x_{0,2} + x_{0,3}) - \delta_0,$$

$$\text{где} \quad |m_0| < 1.051 \cdot 2^{10}, \quad |\delta_0| < 1.998 \cdot 2^{-127},$$

$$|x_{0,1}| < 0.5 \cdot \ln 2 + 1.596 \cdot 2^{-33},$$

$$|x_{0,2}| < 1.595 \cdot 2^{-33}, \quad x_{0,2} \in B_{-33,-77},$$

$$|x_{0,3}| < 1.946 \cdot 2^{-69}, \quad x_{0,3} \in B_{-69,-130}.$$

В силу ограничений по объему статьи мы не приводим здесь доказательства теорем или ограничиваемся только основными идеями доказательств. Полные доказательства будут опубликованы позже.

Следующие три шага сокращения аргумента используют равенство

$$e^x = (1 + v) \cdot e^{x - \ln(1+v)}.$$

При этом мы выбираем значение v так, чтобы величина $x - \ln(1 + v)$ была по возможности малой, а мантисса числа v была короткой.

На втором шаге для поиска величин $1 + v$ и $\ln(1 + v)$ мы используем таблицу W_1, которая создается заранее следующим образом. Для каждого n в диапазоне от -89 до 89 мы находим ближайшее к $n \cdot 2^{-8}$ число вида $\ln(1 + m \cdot 2^{-8})$. Элемент номер n таблицы W_1 содержит плавающее число

$$W_1[n].w_0 = 1 + m \cdot 2^{-8} = 1 + v,$$
$$\text{где} \quad v = m \cdot 2^{-8}$$

и пару чисел расширенной двойной точности, аппроксимирующие $\ln(1 + m \cdot 2^{-8})$:

$$|\ln(1 + m \cdot 2^{-8}) - (W_1[n].w_1 + W_1[n].w_2)| < 2^{-129}.$$

Алгоритм 4.2. Сокращение аргумента. Шаг 2

1: $n_1 \leftarrow R_0(x_{0,1} \cdot 2^8)$

2: $M_1 \leftarrow W_1[n_1].w_0$

3: $x_{1,1} \leftarrow x_{0,1} - W_1[n_1].w_1$ ▷ вычисляется точно, $|x_{1,1}| < 1.175 \cdot 2^{-8}$

4: $x_{1,2} \leftarrow -W_1[n_1].w_2$

В начале второго шага мы находим ближайшее к $x_{0,1}$ число вида $n \cdot 2^{-8}$. Далее с помощью таблицы W_1 мы находим ближайшую к $n \cdot 2^{-8}$ величину вида $\ln(1 + m \cdot 2^{-8})$. Более точно, мы находим пару чисел расширенной двойной точности, аппроксимирующие $\ln(1 + m \cdot 2^{-8})$.

Теорема 4.2. *Пусть x – плавающее число двойной точности такое, что $x_{zero2} \le x \le x_{ovr}$, числа m_0, $x_{0,1}$, $x_{0,2}$ и $x_{0,3}$ получены с помощью алгоритма* 4.1, *а числа M_1, $x_{1,1}$ и $x_{1,2}$ получены с помощью алгоритма* 4.2.

Тогда $x_{1,1}$ и $x_{1,2}$ – плавающие числа расширенной двойной точности, и справедливы следующие равенства и неравенства:

$$x = m_0 \cdot \ln 2 + \ln(M_1) + x_{1,1} + x_{0,2} + x_{1,2} + x_{0,3}$$

$$-\delta_0 - \delta_1, \quad \text{где} \quad |\delta_0| < 1.998 \cdot 2^{-127}, \quad |\delta_1| \lesssim 2^{-129},$$

$$M_1 = 1 + v_1, \quad v_1 = m_1 \cdot 2^{-8}, \quad -75 \le m_1 \le 106,$$

$$|x_{1,1}| < 1.175 \cdot 2^{-8}, \quad x_{1,1} \in B_{-8,-64}, \quad \textit{если} \quad x \ge 2^{-12},$$

$$|x_{1,2}| \le 2^{-65}, \quad x_{1,2} \in B_{-65,-128}.$$

Третий шаг аналогичен второму. На этом шаге мы используем таблицу W_2, содержащую элементы с номерами от -75 до 75. Эта таблица позволяет нам аппроксимировать числа вида $n \cdot 2^{-14}$ числами вида $\ln(1 + m \cdot 2^{-14})$.

Алгоритм 4.3. Сокращение аргумента. Шаг 3

1: $n_2 \leftarrow R_0(x_{1,1} \cdot 2^{14})$

2: $M_2 \leftarrow M_1 \cdot W_2[n_2].w_0$ ▷ вычисляется точно, $0.703 < M_2 < 1.421$, $M_2 \in B_{0,-22}$

3: $x_{2,1} \leftarrow x_{1,1} - W_2[n_2].w_1$ ▷ вычисляется точно, $|x_{2,1}| < 0.675 \cdot 2^{-14}$

4: $x_{2,2} \leftarrow -W_2[n_2].w_2$

Теорема 4.3. *Пусть x – плавающее число двойной точности такое, что $x_{zero2} \le x \le x_{ovr}$, числа m_0, M_2, $x_{0,2}$, $x_{0,3}$, $x_{1,2}$, $x_{2,1}$ и $x_{2,2}$ получены путем последовательного применения алгоритмов* 4.1, 4.2 *и* 4.3. *Тогда $x_{2,1}$ – плавающее число двойной точности, $x_{2,2}$ – плавающее число расширенной двойной точности, и справедливы следующие равенства и неравенства*:

$$x = m_0 \cdot \ln 2 + \ln(M_2) + (x_{2,1} + x_{0,2}) + (x_{1,2} + x_{2,2} + x_{0,3}) - (\delta_0 + \delta_1 + \delta_2), \quad \text{где}$$

$$M_2 = M_1 \cdot (1 + v_2), \quad v_2 = m_2 \cdot 2^{-14},$$

$$-75 \le m_2 \le 75,$$

$$0.703 < M_2 < 1.421, \quad M_2 \in B_{0,-22},$$

$$|x_{2,1}| < 0.675 \cdot 2^{-14},$$

$$x_{2,1} \in B_{-15,-67}, \quad \text{если} \quad x \ge 2^{-15},$$

$$|x_{2,2}| \le 2^{-65}, \quad x_{2,2} \in B_{-65,-128}, \quad |\delta_2| \le 2^{-129}.$$

На четвертом шаге таблицы не применяются, а величины v и $\ln(1+v)$ вычисляются. Для поиска величины v мы используем приближение $x \approx \ln(1 + x)$. Положим $v_3 = 3 \cdot R_{-30}\left(\frac{1}{3} \cdot x_2\right)$, где $x_2 = x_{2,1} + x_{0,2}$. Величина v_3 имеет короткую мантиссу, делится точно на 3 и $x_2 \approx \ln(1 + v_3)$. Для вычисления $\ln(1 + v_3)$ мы используем следующее приближение

$$\ln(1 + v) \approx v - 0.5 \cdot v^2 + \frac{v^3}{3} + v^4 \cdot P_3(v). \qquad (4.3)$$

Полином P_3 был найден с помощью пакета Sollya [6]:

$$P_3(v) = a_0 + v \cdot (a_1 + v \cdot (a_2 + v \cdot a_3)),$$

$$a_0 = \texttt{-0xf.ffffffffffffdp-6},$$

$$a_1 = \texttt{0xc.ccccccccccccc1p-6},$$

$$a_2 = \texttt{-0xa.aaaaaae49a3c83p-6},$$

$$a_3 = \texttt{0x9.249249728e270fdp-6}.$$

Погрешность приближения (4.3) на интервале $[-0.677 \cdot 2^{-14}, 0.677 \cdot 2^{-14}]$ не превышает $1.495 \cdot 2^{-123}$.

Используя (4.3), получаем

$$(x_{2,1} + x_{0,2}) + (x_{0,3} + x_{1,2} + x_{2,2}) = \ln(1 + v_3) + ((x_{2,1} + x_{0,2}) + (x_{0,3} + x_{1,2} + x_{2,2}) - \ln(1 + v_3)) \approx \ln(1 + v_3) + x_{3,1} + x_{3,2}, \quad \text{где}$$

$$x_{3,1} = (x_2 - v_3) + 0.5 \cdot v_3^2 - v_3^2 \cdot v_3',$$

$$x_{3,2} = ((x_{0,3} + x_{1,2}) + x_{2,2}) - v_3^4 \cdot P_3(v_3).$$

Алгоритм 4.4. Сокращение аргумента. Шаг 4

1: $x_2 \leftarrow x_{2,1} + x_{0,2}$ $\quad \triangleright$ x_2 вычисляется точно

2: $v_3' \leftarrow R_{-30}\left(\frac{1}{3} \cdot x_2\right)$

3: $v_3 \leftarrow 3 \cdot v_3' \triangleright |v_3| < 0.677 \cdot 2^{-14}$, $v_3 \in B_{-15,-30}$

4: $x_{3,1} \leftarrow (x_2 - v_3) + 0.5\, v_3^2 - v_3^2 \cdot v_3'$ $\quad \triangleright$ $x_{3,1}$ вычисляется точно, $|x_{3,1}| < 1.210 \cdot 2^{-29}$

5: $x_{3,2} \leftarrow ((x_{0,3} + x_{1,2}) + x_{2,2}) - v_3^4 \cdot P_3(v_3)$

6: $\triangleright$ $|x_{3,2}| < 1.821 \cdot 2^{-61}$, $\Delta(x_{3,2}) < 1.219 \cdot 2^{-123}$

7: $M_3 \leftarrow M_2\,(1 + v_3) \triangleright$ M_3 вычисляется точно, $0.703 < M_3 < 1.421$, $M_3 \in B_{0,-52}$

Теорема 4.4. *Пусть x – плавающее число двойной точности такое, что $x_{zero2} \le x \le x_{ovr}$, числа m_0, M_3, $x_{3,1}$ и $x_{3,2}$ получены путем последовательного применения алгоритмов* 4.1, 4.2, 4.3 *и* 4.4. *Тогда справедливы следующие равенства и неравенства*:

$$x = m_0 \cdot \ln 2 + \ln(M_3) + x_{3,1} + x_{3,2} - \delta_a,$$
$$\text{где} \quad 0.703 < M_3 < 1.421, \quad M_3 \in B_{0,-52},$$

$$|x_{3,1}| < 1.210 \cdot 2^{-29}, \quad |x_{3,2}| < 1.821 \cdot 2^{-61},$$
$$|\delta_a| < 1.436 \cdot 2^{-122}.$$

Из теоремы 4.4 следует, что

$$e^x = 2^{m_0} \cdot M_3 \cdot e^{x_{3,1} + x_{3,2} - \delta_a}. \qquad (4.4)$$

Таким образом, в результате четырех шагов нам удалось сократить аргумент до величины меньшей $1.211 \cdot 2^{-29}$. Так как числа v_1, v_2 и v_3 имеют короткие мантиссы, то M_3 вычисляется точно и является числом двойной точности.

4.2. Вычисление значения функции

Алгоритмы 4.5 и 4.6 завершают вычисление корректно округленного значения экспоненты e^x. Если $x_{dnrm} \le x \le x_{ovr}$, то используется алгоритм 4.5, а если $x_{zero2} \le x < x_{dnrm}$, то используется алгоритм 4.6.

Фактически каждый из этих алгоритмов описывает три алгоритма: один – для округления к ближайшему, второй – для округления к плюс бесконечности и третий – для округления к минус бесконечности (см. комментарии к алгоритмам). Например, в алгоритме 4.5 при округлении к ближайшему используется строка 21, а строки с 22 по 39 не используются. Сами алгоритмы используют при вычислениях только округление к ближайшему (roundTiesToEven).

Теорема 4.5. *Пусть x – число двойной точности. Если $x_{dnrm} \le x \le x_{ovr}$, то алгоритмы* 4.1, 4.2, 4.3, 4.4 *и* 4.5 *вычисляют корректно округленное значение экспоненты e^x. Если $x_{zero2} \le x < x_{dnrm}$, то алгоритмы* 4.1, 4.2, 4.3, 4.4 *и* 4.6 *вычисляют корректно округленное значение экспоненты e^x.*

Полное доказательство теоремы 4.5 будет опубликовано позже.

5. ТЕСТИРОВАНИЕ И ВРЕМЕННЫЕ ХАРАКТЕРИСТИКИ

Предложенные выше алгоритмы были реализованы в виде функции crexp_e() на языке Си. Эта функция была протестирована на компьютере с процессором Core i7-2600K CPU, 3.4 ГГц и операционной системой Linux Fedora 20 Kernel 3.19.8. Для компиляции использовался компилятор gcc, версия 4.8.3, уровень оптимизации 3.

Для тестирования на каждом интервале $[x_{dnrm}, x_{ovr}]$ и $[x_{zero2}, x_{dnrm})$ случайным образом было выбрано по 10,000,000,000 значений аргумента, для которых вычислялось значение функции `crexp_e()` при всех рассматриваемых режимах округления. Дополнительно функция тестировалась при малых значениях аргумента ($|x| < 2^{-28}$) и при значениях аргумента близких к x_{dnrm}. Для тестирования использовались также худшие случаи из [2–4, 7]. Для проверки корректности тестируемой функции использовались функции библиотеки CRlibm, версия 1.0beta4. Все тесты прошли успешно.

При оценке производительности время измерялось в тактах процессора. Реальная частота процессора при измерениях была равна номинальной – 3.4 ГГц. Все измеряемые функции работали в режиме округления к ближайшему.

Мы также рассматривали экспоненциальную функцию ОС Линукс. Более точно, мы использовали функцию `__ieee754_exp()` вместо `exp()`, чтобы избежать динамического вызова. Эта функция вычисляет корректно округленное значение экспоненты при округлении к ближайшему. При этом используется только арифметика двойной точности.

Мы также рассматривали функцию Лаутера [8], которая вычисляет корректно округленное значение экспоненциальной функции при всех рассматриваемых режимах округления. Функция Лаутера использует арифметику расширенной двойной точности, написана на ассемблере и оптимизирована для процессора Intel Itanium.

Алгоритм 4.5. $x_{dnrm} \le x \le x_{ovr}$

1: $\triangleright\ e^x = 2^{m_0} \cdot M_3 \cdot e^{x_{3,1}+x_{3,2}-\delta_a}$
2: $\triangleright\ 0.703 < M_3 < 1.421,\ M_3 \in B_{0,-52}$
3: $\triangleright\ |x_{3,1}| < 1.210 \cdot 2^{-29},\ |x_{3,2}| < 1.821 \cdot 2^{-61},\ |\delta_a| < 1.436 \cdot 2^{-122}$
4: $M_{3,1} \leftarrow R_{-30}(M_3)$
5: $M_{3,2} \leftarrow M_3 - M_{3,1} \triangleright\ M_3 = M_{3,1} + M_{3,2}$
6: $S_1 \leftarrow R_{-62}(x_{3,1}) \triangleright\ S_1$ вычисляется точно, $|S_1| < 1.211 \cdot 2^{-29},\ S_1 \in B_{-29,-62}$
7: $x_3 \leftarrow x_{3,1} + x_{3,2}$
8: $\triangleright\ |x_3| < 1.211 \cdot 2^{-29},\ |\Delta(x_3)| \le 2^{-93}$
9: $S_2 \leftarrow ((x_{3,1} - S_1) + x_{3,2}) + x_3^2 \cdot \left(0.5 + \frac{1}{3!} \cdot x_3\right)$
10: $\triangleright\ |S_2| < 1.985 \cdot 2^{-59},\ |\Delta(S_2)| < 1.850 \cdot 2^{-121}$
11: $\triangleright\ e^{x_{3,1}+x_{3,2}-\delta_a} \approx 1 + S_1 + S_2$
12: $\triangleright\ 2^{-m_0} \cdot e^x \approx (M_{3,1} + M_{3,2}) \cdot (1 + S_1 + S_2) = M_3 + M_{3,1} \cdot S_1 + M_{3,2} \cdot S_1 + M_3 \cdot S_2$
13: $\triangleright\ |M_{3,1} \cdot S_1| < 1.723 \cdot 2^{-29},\ M_{3,1} \cdot S_1 \in B_{-29,-92}$
14: $\triangleright\ (2_{11} - 1) + M_3 \in B_{11,-52}$
15: $(Y, Y_1) \leftarrow Fast2Sum(((2^{11} - 1) + M_3), M_{3,1} \cdot S_1)$
16: $\triangleright\ |M_{3,2} \cdot S_1 + M_3 \cdot S_2| < 1.714 \cdot 2^{-58},\ |\Delta(M_{3,2} \cdot S_1 + M_3 \cdot S_2)| < 1.815 \cdot 2^{-120}.$
17: $Y_2 \leftarrow Y_1 + (M_{3,2} \cdot S_1 + M_3 \cdot S_2)$
18: $\triangleright\ |Y_2| < 1.054 \cdot 2^{-53},\ |\Delta(Y_2)| < 1.227 \cdot 2^{-117}$
19: $\triangleright$ Округленные значения e^x и $2^{m_0} \cdot ((Y - (2^{11} - 1)) + Y_2)$ равны
20: $\triangleright$ Округление к ближайшему
21: $y \leftarrow (Y + Y_2) - (2^{11} - 1)$ $\triangleright\ 0.701 + (2^{11} - 1) < Y + Y_2 < 1.423 + (2^{11} - 1)$
22: $\triangleright$ Округление к $+\infty$
23: $y \leftarrow Y - (2^{11} - 1)$ $\triangleright\ 0.702 \le y \le 1.422$
24: **if** $Y_2 > \mathbf{0}$ then
25: **if** $y \ge 1$ **then**
26: $y \leftarrow y + 2^{-52}$
27: **else**
28: $y \leftarrow y + 2^{-53}$
29: **end if**
30: **end if**
31: $\triangleright$ Округление к $-\infty$ или к 0
32: $y \leftarrow Y - (2^{11} - 1)$
33: **if** $Y_2 < 0$ **then**
34: **if** $y > 1$ **then**
35: $y \leftarrow y - 2^{-52}$
36: **else**
37: $y \leftarrow y - 2^{-53}$
38: **end if**
39: **end if**
40: **return** $y \cdot 2^{m_0}$ $\triangleright$ Восстановление

Алгоритм 4.6. $x_{zero2} \le x < x_{dnrm}$

1: $\triangleright\ e^x = 2^{m_0} \cdot M_3 \cdot e^{x_{3,1}+x_{3,2}-\delta_a}$
2: $\triangleright\ 0.703 < M_3 < 1.421,\ M_3 \in B_{0,-52}$
3: $\triangleright\ |x_{3,1}| < 1.210 \cdot 2^{-29}, x_{3,1} \in B_{-29,-90}$, $|x_{3,2}| < 1.821 \cdot 2^{-61}, |\delta_a| < 1.436 \cdot 2^{-122}$
4: $M_{3,1} \leftarrow R_{-30}(M_3)$
5: $\triangleright\ 0.702 < M_{3,1} < 1.422,\ M_{3,1} \in B_{0,-30}$
6: $M_3' \leftarrow 2^{m_0} \cdot M_3$
7: $M_{3,1}' \leftarrow 2^{m_0} \cdot M_{3,1}$
8: $M_{3,2}' \leftarrow M_3' - M_{3,1}'$
9: $\triangleright\ M_3' = M_{3,1}' + M_{3,2}'$
10: $S_1 \leftarrow R_{-62}(x_{3,1}) \triangleright S_1$ вычисляется точно, $|S_1| < 1.211 \cdot 2^{-29},\ S_1 \in B_{-29,-62}$
11: $x_3 \leftarrow x_{3,1} + x_{3,2}$
12: $\triangleright\ |x_3| < 1.211 \cdot 2^{-29},\ |\Delta(x_3)| \le 2^{-93}$
13: $S_2 \leftarrow ((x_{3,1} - S_1) + x_{3,2}) + x_2^3 \cdot \left(0.5 \cdot \frac{x_3}{3!}\right)$
14: $\triangleright\ |S_2| < 1.985 \cdot 2^{-59},\ |\Delta(S_2)| < 1.850 \cdot 2^{-121}$
15: $Y_1 \leftarrow t_2 + (t_1 + (M_{3,2} \cdot S_1 + M_3 \cdot S_2))$
16: $\triangleright\ e^x \approx M_3' \cdot (1 + S_1 + S_2)$
17: $\triangleright\ 2^{-1011} + e^x \approx 2^{-1011} + M_3' + M_{3,1}' \cdot S_1 + M_{3,2}' \cdot S_1 + M_3' \cdot S_2$
18: $(t_0, t_1) \leftarrow Fast2Sum(M_3', M_{3,1}' \cdot S_1)$
19: $\triangleright\ M_3', M_{3,1}' \cdot S_1, t_0$ и t_1 вычисляются точно
20: $(Y, t_2) \leftarrow Fast2Sum(2^{-1011}, t_0)$
21: $\triangleright\ Y \in B_{-1011,-1074}, Y$ и t_2 вычисляются точно
22: $\triangleright\ e^x \approx (Y - 2^{-1011}) + (t_2 + (t_1 + (M_{3,2}' \cdot S_1 + M_3' \cdot S_2)))$
23: $\triangleright$ Округление
24: $y \leftarrow Y - (2^{-1011} - 2^{-1022})$
25: $\triangleright$ Округление к ближайшему
26: **if** $(t_2 + 2^{-1075}) + (t_1 + (M_{3,2}' \cdot S_1 + M_3' \cdot S_2)) < 0$ **then**
27: $y \leftarrow y - 2^{-1074}$
28: **else if** $(t_2 - 2^{-1075}) + (t_1 + (M_{3,2}' \cdot S_1 + M_3' \cdot S_2)) > 0$ **then**
29: $y \leftarrow y + 2^{-1074}$
30: **end if**
31: $\triangleright$ Округление к $+\infty$
32: **if** $t_2 + (t_1 + (M_{3,2}' \cdot S_1 + M_3' \cdot S_2)) > 0$ **then**
33: $y \leftarrow y + 2^{-1074}$
34: **end if**
35: $\triangleright$ Округление к $-\infty$
36: **if** $t_2 + (t_1 + (M_{3,2}' \cdot S_1 + M_3' \cdot S_2)) < 0$ **then**
37: $y \leftarrow y - 2^{-1074}$
38: **end if**
39: return $y - 2^{-1022}$ $\triangleright$ Восстановление

Таблица 1. Максимальное время (нсек)

	$[x_{dnrm}, x_{ovr}]$	$[x_{zero2}, x_{dnrm})$
crexp_e	141	156
crexp_d	219	220
Lauter	172	4578
CRlibm	735	1023
Linux	76297	6160
Combined	185	157

Таблица 2. Среднее время (нсек)

	$[x_{dnrm}, x_{ovr}]$	$[x_{zero2}, x_{dnrm})$
crexp_e	141	156
crexp_d	188	195
Lauter	172	4578
CRlibm	85	1020
Linux	52	381
Combined	51	157

Для сравнения мы также использовали функцию `crexp_d()`, которая реализует алгоритмы предложенные автором в [4]. Эта функция использует только арифметику двойной точности.

Таблицы 1 и 2 содержат результаты измерений (в тактах процессора). Максимальное время выполнения функций на указанных интервалах приведено в таблице 1. Таблица 2 содержит среднее время выполнения функций. Все измерения для всех функций кроме функции Лаутера были произведены автором. Данные, относящиеся к функции Лаутера, взяты из статьи Лаутера [8].

Как видно из таблицы 1, функция `crexp_e()` обладает наименьшим максимальным временем выполнения на обоих интервалах. Функция Лаутера требует на 22% больше времени на интервале $[x_{dnrm}, x_{ovr}]$ и в 30 раз больше времени на интервале $[x_{zero2}, x_{dnrm})$.

Среднее время выполнения функции `crexp_e()` на интервале $[x_{zero2}, x_{dnrm})$ также меньше, чем у других функций, представленных в таблице, но среднее время выполнения функции `crexp_e()` на интервале $[x_{dnrm}, x_{ovr}]$ больше, чем у функций CRlibm и Линукс. Среднее время можно снизить объединяя функции `crexp_e()` и `__ieee754_exp()` аналогично тому, как это сделано в [4]. Для этого нужно в функции `__ieee754_exp()` заменить вызов функции `__slowexp()` на вызов функции `crexp_e()` на интервале $[x_{dnrm}, x_{ovr}]$. При этом на интервале $[x_{zero2}, x_{dnrm})$ вызывается только функция `crexp_e()`. Полученная таким образом функция

(Combined) имеет лучшее среднее время выполнения на обоих интервалах $[x_{zero2}, x_{dnrm})$ и $[x_{dnrm}, x_{ovr}]$, но максимальное время на интервале $[x_{dnrm}, x_{ovr}]$ увеличится. Отметим, что эта функция может использоваться только в режиме округления к ближайшему.

ИСТОЧНИК ФИНАНСИРОВАНИЯ

Публикация выполнена в рамках государственного задания по проведению фундаментальных исследований по теме “Исследование и реализация программной платформы для перспективных многоядерных процессоров” (FNEF-2022-002).

СПИСОК ЛИТЕРАТУРЫ

1. IEEE Std. 754-2008 – IEEE Standard for Floating-Point Arithmetic. IEEE Std. 2018.
2. *Lefèvre V., Muller J.-M.* Worst cases for correct rounding of the elementary functions in double precision. In Proceedings of the 15th IEEE Symposium on Computer Arithmetic. June 2001. P. 111–118.
3. *Lefèvre V.* Hardest-to-Round Cases – Part 2 // Journées TaMaDi. Lyon. Oct. 2013. [Online]. Available: http://tamadiwiki.enslyon.fr/tamadiwiki/images/c/c1/Lefevre2013.pdf.
4. *Godunov A.* Algorithms for Calculating Correctly Rounded Exponential Function in Double-Precision Arithmetic // IEEE Transactions on Computers. V. 69. № 5. P. 1–12. July 2020. https://doi.org/10.1109/TC.2020.2972901
5. *Daramy-Loirat C., Defour D., de Dinechin F., Gallet M., Gast N., Lauter C.Q., Muller J.-M.* “CR-LIBM, a library of correctly rounded elementary functions in double-precision”, LIP, Research Report, 2006, https://hal-enslyon. archives-ouvertes.fr/ensl-01529804.
6. *Chevillard S., Joldeş M., Lauter C.* Sollya: An Environment for the Development of Numerical Codes. In Mathematical Software – ICMS 2010. P. 28–31, Heidelberg, Germany, September 2010, Springer.
7. *Muller J.-M.* Elementary Functions: Algorithms and Implementation. Birkhauser. 2005.
8. *Lauter C.* A correctly rounded implementation of the exponential function on the Intel Itanium architecture // INRIA, Research Report, RR-5024, 2003. [Online]. Available: https://hal.inria.fr/inria-00071560/document