Информационные процессы, Том 17, № 3, 2017, стр. 213–221

Исследование итерационных алгоритмов вычисления элементарных функций[1]

В.Г. Абрамов, Н.В. Баева, С.Ю. Соловьев, К.Ю. Татунов

МГУ имени М.В.Ломоносова, факультет ВМК, Москва, Россия
vlabr@cs.msu.su, nbaeva@gmail.com, soloviev@glossary.ru, tatunov.k@gmail.com
Поступила в редколлегию 22.09.2017

Аннотация—В работе описывается подход к вычислению элементарных функций посредством специально построенной сетки узлов на полуинтервале $[0.5; 1)$. В качестве модельной задачи, демонстрирующей возможности предложенного подхода, используется функция вычисления двоичного логарифма. Для двух вариантов сетки узлов предложены алгоритмы вычисления логарифмов, а также приводятся и обосновываются оценки сложности этих алгоритмов. Показывается, что в зависимости от свойств сетки оценки средней трудоемкости вычислений могут изменяться весьма значительно.

КЛЮЧЕВЫЕ СЛОВА: функция действительного переменного, вычислительная сложность, численный алгоритм, сетка.

1. ВВЕДЕНИЕ

Изначально вычислительные машины создавались в интересах численных расчетов, в которых во все времена использовались и используются элементарные и/или специальные функции. Скромные возможности первых ЭВМ накладывали серьезные ограничения на методы вычисления таких функций. В частности, требовалось минимизировать количество задействованных в методах констант [1]. Прогресс вычислительной техники предоставил в распоряжение программистов фантастические объемы оперативной памяти и актуализировал задачу разработки новых быстрых методов вычисления функций, не стесненных (в разумных пределах) в части хранения коэффициентов и/или таблиц значений. В настоящей работе исследуется специальный класс быстрых методов вычисления функций действительного переменного:

- в п.2 предлагается общий подход к реализации алгоритмов вычисления элементарных функций;
- в п.3 и п.5 в развитие общего подхода приводятся алгоритмы вычисления двух модельных функций;
- в п.4. и п.6 выводятся оценки трудоемкости модельных алгоритмов, демонстрирующие эффективность подхода.

2. ПОДХОД К ВЫЧИСЛЕНИЮ ЭЛЕМЕНТАРНЫХ ФУНКЦИЙ

Любое положительное число R можно однозначно представить в нормализованном виде $2^p\,x$, где p – целое, а x – вещественное из полуинтервала $[0.5; 1)$. Числа p и x соответственно называются порядком и мантиссой числа R. Для некоторых элементарных функций это представление позволяет свести вычисление значений для произвольного аргумента к вычислению

[1] Работа выполнена за счет гранта РФФИ (проект No. 16-07-00858).

Таблица 1. Первые члены последовательности $\{\rho_i\}$ в шестнадцатиричной системе счисления.

i	$\rho_i - \varepsilon_i$	$\rho_i^{-1} - \delta_i$	i	$\rho_i - \varepsilon_i$	$\rho_i^{-1} - \delta_i$	i	$\rho_i - \varepsilon_i$	$\rho_i^{-1} - \delta_i$
0	0.80000000	2.00000000	11	0.FFE9D2B2	1.00162F39	22	0.FFFFFD3A	1.000002C5
1	0.B504F333	1.6A09E667	12	0.FFF4E91B	1.000B175E	23	0.FFFFFE9D	1.00000162
2	0.D744FCCA	1.306FE0A3	13	0.FFFA747E	1.00058BA0	24	0.FFFFFF4E	1.000000B1
3	0.EAC0C6E7	1.172B83C7	14	0.FFFD3A3B	1.0002C5CC	25	0.FFFFFFA7	1.00000058
4	0.F5257D15	1.0B5586CF	15	0.FFFE9D1C	1.000162E5	26	0.FFFFFFD3	1.0000002C
5	0.FA83B2DB	1.059B0D31	16	0.FFFF4E8E	1.0000B172	27	0.FFFFFFE9	1.00000016
6	0.FD3E0C0C	1.02C9A3E7	17	0.FFFFA747	1.000058B9	28	0.FFFFFFF4	1.0000000B
7	0.FE9E115C	1.0163DA9F	18	0.FFFFD3A3	1.00002C5C	29	0.FFFFFFFA	1.00000005
8	0.FF4ECB59	1.00B1AFA5	19	0.FFFFE9D1	1.0000162E	30	0.FFFFFFFD	1.00000002
9	0.FFA75652	1.0058C86D	20	0.FFFFF4E8	1.00000B17	31	0.FFFFFFFE	1.00000001
10	0.FFD3A751	1.002C605E	21	0.FFFFFA74	1.0000058B	32	0.FFFFFFFF	1.00000000

значений для аргументов из $[0.5;1)$. Скажем, $log_2(R) = p + log_2(x)$ или $\sqrt{R} = p_x\sqrt{x}$, где

$$p_x = \begin{cases} 2^{p/2}, & \text{если } p \text{ – четное или ноль;} \\ 2^{(p-1)/2}\sqrt{2}, & \text{если } p \text{ – нечетное положительное;} \\ 2^{(1-|p|)/2}/\sqrt{2}, & \text{если } p \text{ – нечетное отрицательное.} \end{cases}$$

В дальнейшем изложении в качестве модельного алгоритма используется алгоритм вычисления двоичного логарифма.

Будем рассматривать конечные фрагменты последовательности $\{\rho_i\}_{i=0}^{\infty}$ с общим членом $\rho_i = 2^{-1/2^i}$: $\rho_0 = 1/2$, $\rho_1 = 1/\sqrt{2}$, $\rho_2 = 1/\sqrt[4]{2}$, $\rho_2 = 1/\sqrt[8]{2}$ и т.д. Численно: $\rho_0 = 0.5$, $\rho_1 \approx 0.7071$, $\rho_2 \approx 0.840896$ и т.д. В таблице 1 приводятся первые члены последовательностей $\{\rho_i\}$ и $\{\rho_i^{-1}\}$; для каждого члена упомянутых последовательностей приводятся 32 верных двоичных знака дробной части: $0 \le \varepsilon_i < 2^{-32}$ и $0 \le \delta_i < 2^{-32}$. Приведем формальные свойства последовательности $\{\rho_i\}$, существенные для последующих рассуждений.

Свойство 1. Последовательность $\{\rho_i\}$ является монотонно возрастающей, причем

$$\lim_{i\to\infty} \rho_i = 1.$$

Свойство 2. Если $0 \le k \le m$, то $\rho_k\,\rho_{k+1}\ldots\rho_m = \rho_{k-1}/\rho_m$ и, в частности, $\rho_k = \rho_{k+1}^2$.

Свойство 3. Если $x \in [\rho_{k-1};\rho_k)$, то $x/\rho_k \in [\rho_k;1)$.

Свойство 4. $(1{+}\rho_1)\cdots(1{+}\rho_k) = (1{-}\rho_0)/(1{-}\rho_k)$.

В самом деле, $(1{+}\rho_m)(1{-}\rho_m) = (1{-}\rho_m^2) = (1{-}\rho_{m-1})$ и поэтому:

$$(1{+}\rho_1)\cdots(1{+}\rho_k) = \frac{(1{+}\rho_1)\cdots(1{+}\rho_{k-1})(1{+}\rho_k)(1{-}\rho_k)}{1{-}\rho_k} =$$
$$= \frac{(1{+}\rho_1)\cdots(1{+}\rho_{k-1})(1{-}\rho_{k-1})}{1{-}\rho_k} = \cdots = \frac{(1{+}\rho_1)(1{-}\rho_1)}{1{-}\rho_k} = \frac{1{-}\rho_0}{1{-}\rho_k} = \frac{1}{2(1{-}\rho_k)}.$$

3. МОДЕЛЬНЫЙ АЛГОРИТМ

Зафиксируем некоторое $n > 1$ и соответствующую сетку $0.5 = \rho_0 < \rho_1 < \cdots < \rho_n < 1$. Пусть также $x \in [0.5;1)$. В этом случае известные свойства двоичных алгоритмов трансформируются в два утверждения:

(A) Если $x \in [\rho_{k-1};\rho_k)$, то $log_2(x) = log_2(x/\rho_k) - 2^{-k}$, причем $x/\rho_k \in [\rho_k;1)$.

(B) Если $x \in [\rho_n;1)$, то $log_2(x) \approx 0$, причем $-2^{-n} \le log_2(x) - 0 \le 0$.

Свойства сетки и утверждения (A), (B) позволяют построить алгоритм log2v1 вычисления двоичных логарифмов с точностью 2^{-n}. В описании алгоритма на паскалеподобном алгоритмическом языке считается известной процедура $\mathrm{FEX}(\mathrm{R},\mathrm{p},\mathrm{x})$, которая по заданному положительному R вычисляет его порядок p и мантиссу x; кроме того переменные $n, \rho_1, \ldots, \rho_n$ считаются глобальными, значения которых заданы во внешнем модуле.

```
function log2v1(R : real) : integer;
  var k,p : integer;
        x : real;
begin   FEX(R,p,x);
        R:=0;
        for k:=1 to n do
        if x < ρk then begin x:=x/ρk; R:=R+2^-k end;
        log2v1:=p-R                            end;
```

Комментарий 1. Как следует из свойства 2, оператор "x:=x/ρ_k" вытесняет [2] аргумент x из $[\rho_{k-1}; \rho_k)$ в один из полуинтервалов области $[\rho_k; 1)$.

Комментарий 2. Оператор "for k:=1 to n do" последовательно перебирает полуинтервалы $[\rho_{k-1}; \rho_k)$, а свойство 3 гарантирует достаточность однократного перебора для вытеснения аргумента x в $[\rho_n; 1)$.

Комментарий 3. Операторы "R:=0" и "R:=R+2^{-k}" фактически конструируют из нулей и единиц искомый результат в двоичной системе счисления. Это обстоятельство позволяет отнести алгоритм log2v1 к классу алгоритмов "цифра за цифрой" [3].

Комментарий 4. С точки зрения временной сложности-в-худшем-случае оператор "for k:=1 to n do" обеспечивает оценку $O(n)$.

4. СЛОЖНОСТЬ МОДЕЛЬНОГО АЛГОРИТМА В СРЕДНЕМ

Выберем некоторое целое положительное число n. В настоящем разделе будем приписывать индекс f полуинтервалу $[a; b) \subset [0.5; 1)$, если для заданной сетки $\rho_0 < \rho_1 < \cdots < \rho_n < 1$ и для любого $x \in [a; b)$ алгоритм log2v1 использует ровно f делений вида $\bullet/\rho_k$.

Замечание 1. Преобразование $\bullet/\rho_k$ взаимно однозначным образом отображает полуинтервал $[\rho_{k-1}; \rho_k)$ на $[\rho_k; 1)$, и поэтому любой индексированный полуинтервал $[a; b)_f \subset [\rho_k; 1)$ имеет прообраз $[\rho_k a; \rho_k b)_{f+1} \subset [\rho_{k-1}; \rho_k)$.

Построим разбиения M_k множеств $[\rho_k; 1)$ индексированными полуинтервалами:

(a) $M_n \stackrel{def}{=} \Big\{ [\rho_n; 1)_0 \Big\}$;

(b) $M_k \stackrel{def}{=} M_{k+1} \cup \Big\{ [\rho_{k+1} a; \rho_{k+1} b)_{f+1} \,|\, [a; b)_f \in M_{k+1} \Big\}, \quad k = n-1, n-2, \ldots, 0.$

Корректность приведенного определения в части (a) очевидна, а корректность в части (b) следует из замечания 1. В качестве примера приведем разбиения для n=3:

$$M_3 = \Big\{ [\rho_3; 1)_0 \Big\}, \quad M_2 = M_3 \cup \Big\{ [\rho_2; \rho_3)_1 \Big\}, \quad M_1 = M_2 \cup \Big\{ [\rho_1; \rho_2\rho_3)_2,\, [\rho_2\rho_3; \rho_2)_1 \Big\},$$
$$M_0 = M_1 \cup \Big\{ [\rho_0; \rho_1\rho_2\rho_3)_3,\, [\rho_1\rho_2\rho_3; \rho_1\rho_2)_2,\, [\rho_1\rho_2; \rho_1\rho_3)_2,\, [\rho_1\rho_3; \rho_1)_1 \Big\}.$$

С каждым разбиением M_k свяжем средневзвешенное $\sigma_n(k)$:

$$\sigma_n(k) \stackrel{def}{=} \sum_{[a;b)_f \in M_k} (b-a) * f,$$

и приведем для него более компактное аналитическое выражение.

$$\begin{aligned}\sigma_n(k) &= \sum_{[a;b)_f\in M_{k+1}}(b{-}a)*f \;+ \sum_{[a;b)_f\in M_k\setminus M_{k+1}}(b{-}a)*f \;= \\ &= \sigma_n(k{+}1) \;+ \sum_{[a;b)_f\in M_{k+1}}(\rho_{k+1}b{-}\rho_{k+1}a)*(f{+}1) \;= \\ &= \sigma_n(k{+}1) \;+\rho_{k+1}*\sum_{[a;b)_f\in M_{k+1}}(b{-}a)*f \;+\rho_{k+1}*\sum_{[a;b)_f\in M_{k+1}}(b{-}a) \;= \\ &= \sigma_n(k{+}1) \;+\rho_{k+1}*\sigma_n(k{+}1) \;+\rho_{k+1}*(1{-}\rho_{k+1})\end{aligned}$$

То есть

$$\sigma_n(k) = \begin{cases}0, & \text{если } k=n;\\ (1{+}\rho_{k+1})\sigma_n(k{+}1)+\rho_{k+1}*(1{-}\rho_{k+1}), & \text{если } k<n,\end{cases}$$

и далее:

$$\begin{aligned}\sigma_n(0)-\sigma_{n-1}(0) &= (1{+}\rho_1)\Big(\sigma_n(1)-\sigma_{n-1}(1)\Big) = (1{+}\rho_1)(1{+}\rho_2)\Big(\sigma_n(2)-\sigma_{n-1}(2)\Big) = \cdots = \\ &= (1{+}\rho_1)(1{+}\rho_2)\cdots(1{+}\rho_{n-1})\Big(\sigma_n(n{-}1)-\sigma_{n-1}(n{-}1)\Big) = \\ &= (1{+}\rho_1)(1{+}\rho_2)\cdots(1{+}\rho_{n-1})\Big(\rho_n(1-\rho_n)-0\Big) = \qquad \text{/см. свойство 4/} \\ &= \frac{1}{2}\frac{1}{1{-}\rho_{n-1}}\rho_n(1-\rho_n) \;=\; \frac{\rho_n}{2}\frac{1{-}\rho_n}{1{-}\rho_n^2} \;=\; \frac{1}{2}\frac{\rho_n}{1+\rho_n}\end{aligned}$$

Таким образом

$$\sigma_n(0) \;=\; \sigma_{n-1}(0)+\frac{1}{2}\frac{\rho_n}{1+\rho_n} \;=\; \sigma_{n-2}(0)+\frac{1}{2}\left(\frac{\rho_{n-1}}{1+\rho_{n-1}}+\frac{\rho_n}{1+\rho_n}\right) \;=\; \cdots \;=\; \frac{1}{2}\sum_{k=1}^{n}\frac{\rho_k}{1+\rho_k}.$$

Вывод 1. Если алгоритм log2v1 тестируется в серии испытаний, в которой аргументы равномерно распределены на $[0.5;1)$, то число E_n – среднее количество делений вида $\bullet/\rho_k$, приходящихся на одно вычисление log2v1(x), имеет предел по вероятности:

$$E_n \;\xrightarrow{P}\; 2{*}\sigma_n(0) \;=\; \sum_{k=1}^{n}\frac{\rho_k}{1+\rho_k}. \tag{1}$$

Если n достаточно велико, то $\rho_n/(1{+}\rho_n)\approx 1/2$, и поэтому $E_{n+2}\approx E_n+1$.

Замечание 2. В формуле (1) множитель 2 есть нормировочный коэффициент – величина обратная

$$\sum_{[a;b)_f\in M_k}(b{-}a) = 1{-}0.5 = 0.5.$$

Замечание 3. Из построения формулы для $\sigma_n(0)$ следует, что

$$\sigma_n(k) = \begin{cases}0, & \text{если } k=n;\\ (1{-}\rho_k)\,\sum_{i=k+1}^{n}\dfrac{\rho_i}{1+\rho_i}, & \text{если } k<n,\end{cases}$$

5. МОДИФИЦИРОВАННЫЙ МОДЕЛЬНЫЙ АЛГОРИТМ

Зададимся вопросом о возможности усовершенствования алгоритма log2v1 с целью снижения его трудоемкости. Положительный ответ на этот вопрос связан с модификацией сетки. Рассмотрим набор узлов $\rho_0, \rho_1, \ldots, \rho_{n-2}, \rho_{n-1}, \rho_n$, для фиксированного $n \geq 3$. Добавим в этот набор в качестве узлов числа $\mu_1{=}\rho_1\,\rho_2,\ \ldots,\ \mu_{n-1}{=}\rho_{n-1}\,\rho_n$:

$$\rho_0 < \mu_1 < \rho_1 < \cdots < \rho_{n-2} < \mu_{n-1} < \rho_{n-1} < \rho_n. \tag{2}$$

Для случая $n = 3$ сетка (2) имеет вид

$$\rho_0 < \mu_1 < \rho_1 < \mu_2 < \rho_2 < \rho_3, \qquad \text{где}\ \ \mu_1 = \rho_1\,\rho_2,\ \mu_2 = \rho_2\,\rho_3. \tag{3}$$

Расширенная таким образом сетка обладает двумя дополнительными свойствами:
Свойство 3C. Если $x \in [\rho_{k-1}; \mu_k)$, то $x/\mu_k \in [\rho_{k+1}; 1)$.
Свойство 3D. Если $x \in [\mu_k; \rho_k)$, то $x/\rho_k \in [\rho_{k+1}; 1)$.
Применительно к двоичным логарифмам новые свойства трансформируются в следующие утверждения:

(C) Если $x \in [\rho_{k-1}; \mu_k)$, то $log_2(x) = log_2(x/\mu_k) - 2^{-k} - 2^{-k-1}$, причем $x/\mu_k \in [\rho_{k+1}; 1)$.
(D) Если $x \in [\mu_k; \rho_k)$, то $log_2(x) = log_2(x/\rho_k) - 2^{-k}$, причем $x/\rho_k \in [\rho_{k+1}; 1)$.
(E) Если $x \in [\rho_{n-1}; \rho_n)$, то $log_2(x) = log_2(x/\rho_n) - 2^{-n}$, причем $x/\rho_n \in [\rho_n; 1)$.
(F) Если $x \in [\rho_n; 1)$, то $log_2(x) \approx 0$.

В свою очередь учет утверждений (C)–(D) позволяет построить алгоритм log2v2, представляющий собой модифицированную версию алгоритма log2v1.

```
function log2v2(R : real) : integer;
  var k,p : integer;
        x : real;
begin   FEX(R,p,x);
        R:=0; k:=0;
        while k < n−1 do begin
           k:=k+1;
           if x < μ_k then begin x:=x/μ_k; k:=k+1; R:=R+2^{-k}+2^{-k-1} end;
           if x < ρ_k then begin x:=x/ρ_k; k:=k+1; R:=R+2^{-k}          end;
        end;
        if x < ρ_n then begin x:=x/ρ_n; R:=R+2^{-n} end;
        log2v2:=p-R                                                    end;
```

Комментарий 5. Как следует из утверждений (C) и (D), операторы “`x:=x/`μ_k” и “`x:=x/`ρ_k” вытесняют аргумент x из $[\rho_{k-1}; \rho_k)$ в один из полуинтервалов области $[\rho_{k+1}; 1)$, то есть следующий по порядку полуинтервал $[\rho_k; \rho_{k+1})$ цикл “`while`” должен пропустить, и эту операцию обеспечивают дополнительные операторы “k`:=`k`+1;`”.

Комментарий 6. Фрагмент “k`:=0; while` k `<` $n-1$ `do begin` k`:=`k`+1;`” обеспечивает последовательный перебор пар полуинтервалов $[\rho_{k-1}; \mu_k)$ и $[\mu_k; \rho_k)$, а свойство 3 гарантирует достаточность однократного перебора полуинтервалов для вытеснения аргумента x в $[\rho_n; 1)$.

Комментарий 7. Алгоритмы log2v1 и log2v2, помимо достаточно компактных кодов, предполагают наличие таблиц для величин ρ_i и μ_i, которые соответственно содержат $n{+}1$ и $n{-}1$ вещественных чисел.

Комментарий 8. С точки зрения временной сложности-в-худшем-случае алгоритма log2v2 фрагмент “k`:=0; while` k `<` $n-1$ `do begin` k`:=`k`+1;`” обеспечивает оценку $O(n)$.

6. СЛОЖНОСТЬ МОДИФИЦИРОВАННОГО МОДЕЛЬНОГО АЛГОРИТМА В СРЕДНЕМ

Покажем, что по сравнению с алгоритмом log2v1 алгоритм log2v2 сокращает средний объем вычислений. В настоящем разделе будем приписывать индекс f полуинтервалу $[a;b) \subset [0.5;1)$, если для заданной сетки (2) и для любого $x \in [a;b)$ алгоритм log2v2 использует ровно f делений вида $\bullet/\rho_k$ или $\bullet/\mu_k$.

Замечание 4. Преобразования $\bullet/\mu_k$ и $\bullet/\rho_k$ взаимно однозначным образом отображают полуинтервалы $[\rho_{k-1};\mu_k)$ и $[\mu_k;\rho_k)$ на один и тот же полуинтервал $[\rho_{k+1};1)$, а значит любой индексированный полуинтервал $[a;b)_f \subset [\rho_{k+1};1)$ имеет в качестве прообразов два полуинтервала $[\rho_k a;\rho_k b)_{f+1}$ и $[\mu_k a;\mu_k b)_{f+1}$.

Построим разбиения N_k множеств $[\rho_k;1)$ индексированными полуинтервалами:

(a) $N_n \overset{def}{=} \Big\{ [\rho_n;1)_0 \Big\}$;

(b) $N_{n-1} \overset{def}{=} \Big\{ [\rho_{n-1};\rho_n)_1 \Big\}$;

(c) $$N_k \overset{def}{=} N_{k+1} \cup \Big\{ [\mu_{k+1}a;\mu_{k+1}b)_{f+1} \,|\, [a;b)_f{\in}N_{k+2} \Big\} \cup \Big\{ [\rho_{k+1}a;\rho_{k+1}b)_{f+1} \,|\, [a;b)_f{\in}N_{k+2} \Big\}, \qquad k=n{-}2,\dots,0.$$

Корректность приведенного определения в частях (a) и (b) очевидна, а корректность в части (c) следует из замечания 4. В качестве примера приведем разбиения для сетки (3) :

$$N_3 = \Big\{ [\rho_3;1)_0 \Big\}, \quad N_2 = N_3 \cup \Big\{ [\rho_2;\rho_3)_1 \Big\}, \quad N_1 = N_2 \cup \Big\{ [\rho_1;\mu_2)_2 \Big\} \cup \Big\{ [\mu_2;\rho_2)_1 \Big\},$$
$$N_0 = N_1 \cup \Big\{ [\rho_0;\mu_1\rho_3)_2,\, [\mu_1\rho_3;\mu_1)_1 \Big\} \cup \Big\{ [\mu_1;\rho_1\rho_3)_2,\, [\rho_1\rho_3;\rho_1)_1 \Big\}.$$

С каждым разбиением N_k свяжем средневзвешенное $\omega_n(k)$:

$$\omega_n(k) \overset{def}{=} \sum_{[a;b)_f \in N_k} (b{-}a){*}f,$$

и приведем для него рекуррентное выражение.

$$\begin{aligned}
\omega_n(k) &= \sum_{[a;b)_f \in N_{k+1}} (b{-}a){*}f \;+ \sum_{[a;b)_f \in N_k \setminus N_{k+1}} (b{-}a){*}f \;= \\
&= \omega_n(k{+}1) + \sum_{[a;b)_f \in N_{k+2}} \Big(\mu_{k+1}(b{-}a){*}(f{+}1) + \rho_{k+1}(b{-}a){*}(f{+}1) \Big) = \\
&= \omega_n(k{+}1) + \big(\mu_{k+1}{+}\rho_{k+1}\big) \sum_{[a;b)_f \in N_{k+2}} (b{-}a){*}(f{+}1) = \\
&= \omega_n(k{+}1) + \big(\mu_{k+1}{+}\rho_{k+1}\big) \Big(\sum_{[a;b)_f \in N_{k+2}} (b{-}a){*}f + \sum_{[a;b)_f \in N_{k+2}} (b{-}a) \Big) = \\
&= \omega_n(k{+}1) + \big(\mu_{k+1}{+}\rho_{k+1}\big) \big(\omega_n(k{+}2) {+} 1 {-} \rho_{k+2} \big) = \\
&= \omega_n(k{+}1) + \big(\mu_{k+1}{+}\rho_{k+1}\big)\, \omega_n(k{+}2) + \rho_{k+1}\,(1{+}\rho_{k+2})\,(1{-}\rho_{k+2}) = \ldots = \\
&= \omega_n(k{+}1) + \big(\mu_{k+1}{+}\rho_{k+1}\big)\, \omega_n(k{+}2) + \rho_{k+1} - \rho_k
\end{aligned}$$

Эквивалентное преобразование “$= \ldots =$” основывается на равенстве $\rho_{k+1}\,(1{+}\rho_{k+2})\,(1{-}\rho_{k+2}) = \rho_{k+1}\,(1{-}\rho_{k+1}) = \rho_{k+1}{-}\rho_k$. С учетом значений для случаев $k=n$ и $k=n{-}1$ формула для $\omega_n(k)$ принимает следующий законченный вид:

$$\omega_n(k) = \begin{cases} 0, & \text{если } k=n; \\ \rho_n{-}\rho_{n-1}, & \text{если } k=n{-}1; \\ \omega_n(k{+}1) + (\mu_{k+1}{+}\rho_{k+1})\,\omega_n(k{+}2) + \rho_{k+1}{-}\rho_k, & \text{если } k<n{-}1. \end{cases}$$

Лемма 1. *Для любого* $n, n > 2$, *и любого* $k, 1 \le k \le n$, *выполняется равенство*

$$\omega_n(k) - \omega_{n-1}(k) = \Delta_n(k), \quad \textit{где} \tag{4}$$

$$\Delta_n(k) = (-1)^{n+1}\frac{1-\rho_n}{\rho_n}\Delta'_n(k) \quad \textit{и} \quad \Delta'_n(k) = \begin{cases} 0, & \textit{если } k = n; \\ (1+\rho_{k+1})\Delta'_n(k+1) + (-1)^k\rho_k, & \textit{если } k < n. \end{cases}$$

Доказательство леммы проведем индукцией по k, полагая n фиксированным.

1. Пусть $k = n-1$, тогда
$\omega_n(k)-\omega_{n-1}(k) \;=\; \rho_n-\rho_{n-1} - 0 \;=\; \rho_n-\rho_{n-1}$,
$\Delta'(k) = (1+\rho_n)\Delta'_n(n) + (-1)^{n-1}\rho_{n-1} \;=\; (-1)^{n-1}\rho_{n-1}$,
$\Delta(k) = (-1)^{n+1}\dfrac{1-\rho_n}{\rho_n}(-1)^{n-1}\rho_{n-1} = \dfrac{1-\rho_n}{\rho_n}\rho_n^2 \;=\; \rho_n-\rho_n^2 \;=\; \rho_n-\rho_{n-1}$,

то есть равенство (4) справедливо для $k = n-1$.

2. Пусть $k = n-2$, тогда
$\omega_n(k)-\omega_{n-1}(k) =$
$= \omega_n(n-1)+(\cdot)\omega_n(n) + \rho_{n-1}-\rho_{n-2} - (\rho_{n-1}-\rho_{n-2}) =$
$= \rho_n-\rho_{n-1}+(\cdot)\omega_n(n) \;=\; \rho_n-\rho_{n-1}$,
$\Delta'(k) = (1+\rho_{n-1})\Delta'_n(n-1) + (-1)^{n-2}\rho_{n-2} =$
$= (1+\rho_{n-1})\Big((\cdot)\Delta'_n(n) + (-1)^{n-1}\rho_{n-1}\Big) + (-1)^{n-2}\rho_{n-2} =$
$= (1+\rho_{n-1})(-1)^{n-1}\rho_{n-1} + (-1)^{n-2}\rho_{n-2} =$
$= (-1)^{n-1}\Big((\rho_{n-1}+\rho_{n-2}) - \rho_{n-2}\Big) \;=\; (-1)^{n-1}\rho_{n-1}$,
$\Delta(k) = (-1)^{n+1}\dfrac{1-\rho_n}{\rho_n}(-1)^{n-1}\rho_{n-1} = \rho_n-\rho_{n-1}$,

то есть равенство (4) справедливо для $k = n-2$.

3. Предположим равенство (4) справедливо для $k{=}m{+}1, m{+}2, \ldots, n{-}1$, тогда
$\omega_n(m)-\omega_{n-1}(m-1) =$
$= +\omega_n(m+1)+\rho_{m+1}(1+\rho_{m+2})\,\omega_n(m+2) + \rho_{m+1} - \rho_m -$
$-\omega_{n-1}(m+1)-\rho_{m+1}(1+\rho_{m+2})\,\omega_{n-1}(m+2) - \rho_{m+1} + \rho_m =$ /по предположению/
$= \Delta_n(m+1)+\rho_{m+1}(1+\rho_{m+2})\Delta_n(m+2) =$
$= \Delta_n(m+1)+\rho_{m+1}\Big((1+\rho_{m+2})\Delta_n(m+2) + (-1)^{m+1}\rho_{m+1} - (-1)^{m+1}\rho_{m+1}\Big) =$
$= \Delta_n(m+1)+\rho_{m+1}\Delta_n(m+1) - (-1)^{m+1}\rho_{m+1}^2 =$
$= (1+\rho_{m+1})\Delta_n(m+1) + (-1)^m\rho_m \;=\; \Delta_n(m)$.
Лемма доказана.

Лемма 2.

$$\lim_{n\to\infty}\Big(\omega_n(0)-\omega_{n-1}(0)\Big) = \frac{1}{6}.$$

Доказательство леммы 2 состоит из двух частей.

1. $\Delta'_3(0) = +(1+\rho_1)(1+\rho_2)\rho_2 - (1+\rho_1)\rho_1 + \rho_0$,
$\Delta'_4(0) = -(1+\rho_1)(1+\rho_2)(1+\rho_3)\rho_3 + (1+\rho_1)(1+\rho_2)\rho_2 - (1+\rho_1)\rho_1 + \rho_0 =$
$= -(1+\rho_1)(1+\rho_2)(1+\rho_3)\rho_3 + \Delta'_3(0)$.

В общем случае $\Delta'_n(0) = (-1)^{n+1}(1+\rho_1)\cdots(1+\rho_{n-1})\rho_{n-1} + \Delta'_{n-1}(0) \;\Rightarrow$ /см. свойство 4/

$$\Delta'_n(0) = \Delta'_{n-1}(0) + (-1)^{n+1}\frac{1}{2}\,\frac{\rho_{n-1}}{1-\rho_{n-1}} \;\Rightarrow \qquad \text{/см. лемму 1/}$$

$$\Delta_n(0) = \frac{1-\rho_n}{\rho_n}\left((-1)^{n+1}\Delta'_{n-1}(0) + \frac{1}{2}\frac{\rho_{n-1}}{1-\rho_{n-1}}\right) =$$
$$= \frac{1-\rho_n}{\rho_n}\left((-1)^{n+1}\Big((-1)^n\frac{\rho_{n-1}}{1-\rho_{n-1}}\Delta_{n-1}(0)\Big) + \frac{1}{2}\frac{\rho_{n-1}}{1-\rho_{n-1}}\right) =$$
$$= \frac{1-\rho_n}{\rho_n}\frac{\rho_n^2}{1-\rho_n^2}\left(-\Delta_{n-1}(0) + \frac{1}{2}\right) = \frac{\rho_n}{1+\rho_n}\left(-\Delta_{n-1}(0) + \frac{1}{2}\right)$$

2. Положим $\chi_n = \Delta_n(0) - 1/6$.

При этом формула, полученная для $\Delta_n(0)$, трансформируется в

$$\chi_n = -\frac{\rho_n}{1+\rho_n}\chi_{n-1} - \frac{1}{6}\frac{1-\rho_n}{1+\rho_n},$$

а поскольку

$$\frac{1}{3} \le \frac{\rho_n}{1+\rho_n} < \frac{1}{2} \quad \text{и последовательность} \quad \left\{\frac{1-\rho_n}{1+\rho_n}\right\} \text{ монотонно убывает до 0,}$$

$$\text{то} \quad \lim_{n\to\infty}\chi_n = 0 \qquad \Rightarrow \qquad \lim_{n\to\infty}\Delta_n = \frac{1}{6}.$$

Лемма доказана.

Вывод 2. Если алгоритм log2v2 тестируется в серии испытаний, в которой аргументы равномерно распределены на $[0.5;1)$, то число E'_n – среднее количество делений вида $\bullet/\rho_k$ и $\bullet/\mu_k$, приходящихся на одно вычисление log2v2(x), имеет предел по вероятности:

$$E'_n \xrightarrow{P} 2{*}\omega_n(0) = \sum_{k=1}^{n} 2\Big(\omega_i(0) - \omega_{i-1}(0)\Big).$$

Если n достаточно велико, то из леммы 2 следует, что $2\Big(\omega_i(0) - \omega_{i-1}(0)\Big) \approx 1/3$ и, поэтому $E'_{n+3} \approx E_n + 1$. В таблице 2 приводятся сложности-в-среднем алгоритмов log2v1 и log2v2 для некоторых n.

Таблица 2. Сравнительные характеристики трудоемкости алгоритмов log2v1 и log2v2.

n	$2\sigma_n(0)$	$2\omega_n(0)$	Разность
6	2.83	2.02	0.81
7	3.33	2.35	0.98
8	3.83	2.68	1.14
9	4.33	3.02	1.31
10	4.83	3.35	1.48
11	5.33	3.68	1.64

n	$2\sigma_n(0)$	$2\omega_n(0)$	Разность
12	5.83	4.02	1.81
13	6.33	4.35	1.98
14	6.83	4.68	2.14
15	7.33	5.02	2.31
16	7.83	5.35	2.48
17	8.33	5.68	2.64

. . .

n	$2\sigma_n(0)$	$2\omega_n(0)$	Разность
30	14.83	10.02	4.81
31	15.33	10.35	4.98
32	15.83	10.68	5.14
33	16.33	11.02	5.31
34	16.83	11.35	5.48
35	17.33	11.68	5.64

. . .

Замечание 5. Расчеты показывают, что при достаточно больших n

$$2\,\sigma_n(0) \approx -0.172 + n/2,$$
$$2\,\omega_n(0) \approx +0.017 + n/3.$$

Замечание 6. Операции деления вещественных чисел $\bullet/\rho_k$ и $\bullet/\mu_k$, задействованные в алгоритмах log2v1 и log2v2, можно заменить умножениями, если заранее запасти константы ρ_k^{-1} (см. таблицу 1) и μ_k^{-1}. Все остальные операции реализуются быстрыми командами логического типа. Выводы 1 и 2 при этом остаются в силе с поправкой на замену фразы "среднее количество делений" на фразу "среднее количество умножений".

7. ЗАКЛЮЧЕНИЕ

Переход от log2v1 к log2v2 состоял в привлечении в качестве узлов некоторых произведений чисел ρ_k. Если же к сетке $\rho_0 < \rho_1 < \cdots < \rho_n$ добавить всевозможные произведения[1] узлов $\rho_1, \ldots, \rho_n$, то вновь построенная сетка из 2^n узлов позволит находить первые n двоичных разрядов числа $log_2(x)$ вообще без использования операций деления/умножения над вещественными числами. При таком подходе расширенная сетка фактически играет роль специфической таблицы значений, а задача вычисления логарифма сводится к поиску в таблице. Намеченный таблично-ориентированный метод вычисления функций действительного переменного нуждается в отдельном исследовании.

СПИСОК ЛИТЕРАТУРЫ

1. Березин И.С., Жидков Н.П. *Методы вычислений.* Т. I. М.: Наука, 1966.

2. Соловьев С.Ю. Алгоритм вычисления логарифмов методом вытеснения. *Вестн. Моск. ун-та сер. 15 Вычисл. матем. и киберн.*, 2013, No.2, стр. 38-43. $http://www.park.glossary.ru/$?22

3. Байков В.Д., Смолов В.Б. *Специализированные процессоры: Итерационные алгоритмы и структуры.* М.: Радио и связь, 1985.

The study of iterative algorithms for calculating elementary functions

V.G. Abramov, N.V. Baeva, S.Y. Soloviev, K.Y. Tatunov

In this paper we describe the approach to the calculation of elementary functions with the help of a specially constructed mesh of nodes on the half-open interval $[0.5; 1)$. We use the function of computing the binary logarithm as model task to demonstrate the possibility of the proposed approach. We proposed two variants of mesh for calculating the log_2, and we give and justify the complexity of these algorithms. Also we show that, depending on the properties of the mesh, estimates on the average of the complexity can vary considerably.

KEYWORDS: function of a real variable, computational complexity, numerical algorithms, mesh.

[1] Произведения без кратных вхождений сомножителей