УДК 519.683 (авторская копия для www.park.glossary.ru)

С. Ю. Соловьев[1]

АЛГОРИТМ ВЫЧИСЛЕНИЯ ЛОГАРИФМОВ МЕТОДОМ ВЫТЕСНЕНИЯ

Предлагается метод разложения вещественных чисел на сомножители специального вида, каждый из которых, в свою очередь, допускает аналогичное разложение. Описывается связь метода с двоичным представлением вещественных чисел. Обсуждаются вопросы программной реализации метода для задачи вычисления натурального логарифма. Рассматриваются подходы к обобщению метода.

Ключевые слова: логарифм, метод, редукция, погрешность, алгоритм.

1. Введение. Любое положительное число X можно однозначно представить в нормализованном виде $2^P U$, где P – целое, а U – вещественное из полуинтервала $[0.5, 1)$; P называется порядком, U – мантиссой. Применительно к логарифму это означает существование представления $\ln(X) = \ln(U) - P\ln(0.5)$. Иначе говоря, для вычисления любого логарифма достаточно знать алгоритм его вычисления на $[0.5, 1)$.

2. Метод вытеснения. Приведем цепочку формальных свойств, обосновывающих способ редукции (см. [1]) аргумента – формулу (1), применимую для вычисления значений некоторых элементарных функций. Свойства относятся к числам из полуинтервала $[0.5, 1)$ и в силу своей простоты приводятся без доказательств. Прежде всего введем специальные обозначения для трех числовых последовательностей:

$$A_n = \frac{2^n-1}{2^n}, \qquad B_n = \left(\frac{2^n-1}{2^n}\right)^2 = A_n^2, \qquad C_n = \frac{2^n-2}{2^n-1}, \qquad \text{где } \; n = 1, 2, \ldots .$$

Последовательность A_n порождает семейство нумерованных непересекающихся полуинтервалов $I_2, I_3, \ldots$ вида $I_n = [A_{n-1}, A_n)$. В совокупности эти полуинтервалы образуют разбиение (см. рис. 1) множества $[0.5, 1)$, то есть $[0.5, 1) = I_2 \cup I_3 \cup \ldots$. Для полуинтервала $[A_n, 1)$ будем использовать обозначение: $I_{n+} = I_{n+1} \cup I_{n+2} \cup \ldots$.

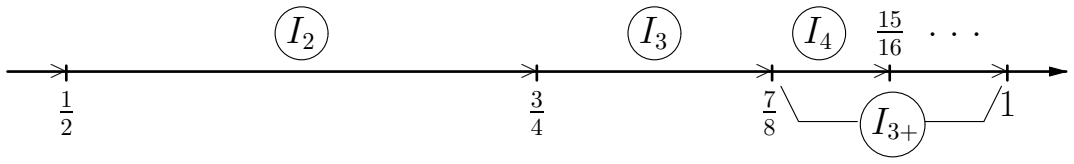

Рис. 1 Семейство полуинтервалов $I_2, I_3, \ldots$

[1] Факультет ВМК МГУ, проф., д.ф.-м.н., e–mail: soloviev@glossary.ru

Свойство 1. $A_{n-1} < B_n < C_n < A_n$ для всех $n \geq 2$ (см. рис. 2*a*).

Любое число x из $[0.5, 1)$ принадлежит ровно одному полуинтервалу I_n, а значит, существует однозначная функция $z\colon [0.5, 1) \to \{2, 3, ...\}$, принимающая значение n, если $x \in I_n$. Например, $z(0.625) = 2$, $z(0.828125) = 3$.

Пусть $x \in I_z = [A_{z-1}, B_z) \cup [B_z, A_z)$. Рассмотрим число x/A_z.

Свойство 2. Если $x \in [A_{z-1}, B_z)$, то $x/A_z \in [C_z, A_z) \subset I_z$ (см. рис. 2*b*).

Свойство 3. Если $x \in [B_z, A_z)$, то $x/A_z \in [A_z, 1) = I_{z+}$ (см. рис. 2*b*).

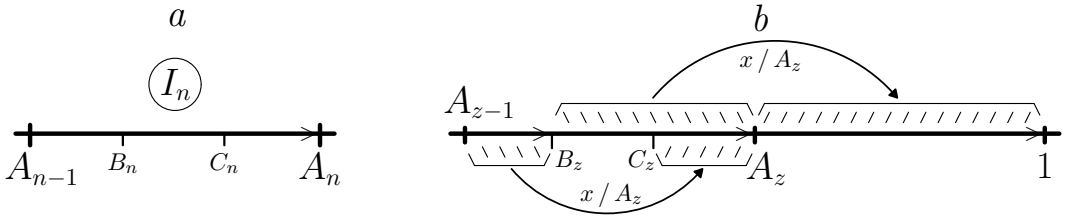

Рис. 2 Свойства 1, 2 и 3 как основания свойства 4 и формулы (1)

Как следует из свойств 2 и 3, преобразование полуинтервала I_z посредством деления его чисел на A_z приводят к двойственному результату:
– числа из $[B_z, A_z)$ вытесняются в полуинтервал I_{z+}, а
– числа из $[A_{z-1}, B_z)$ остаются в I_z, но свойство 1 гарантирует, что повторное деление на A_z все-таки вытеснит их в I_{z+}.

Свойство 4. Если $x \in I_z$, то справедливо равенство

$$x = \begin{cases} (x/A_z^2)A_zA_z, & \text{если } x < B_z, \\ (x/A_z)A_z \ - & \text{иначе,} \end{cases} \tag{1}$$

причем $A_z \in I_{z+1}$, $x/A_z^2 \in I_{z+}$ для $x < B_z$ и $x/A_z \in I_{z+}$ – иначе.

Логарифмируя правую и левую части равенства (1), имеем

Свойство 5. Если $x \in I_z$, то справедливы следующие равенство и связанные с ним соотношения:

$$\ln(x) = \begin{cases} \ln(x/A_z^2) + 2\ln(A_z), & \text{если } x < B_z; \\ \ln(x/A_z) + \ \ln(A_z) - & \text{иначе;} \end{cases} \tag{2}$$

$$\left.\begin{array}{r} A_z \in I_{z+1}, \\ \text{если } x < B_z, \text{ то } x/A_z^2 \in I_{z+}, \\ \text{если } x \geq B_z, \text{ то } x/A_z \in I_{z+} \end{array}\right\}. \tag{3}$$

Другими словами, формула (2) сводит вычисление $ln(x)$ для $x \in I_z$:
либо к паре подзадач $\ln(A_z)$ (для $A_z \in I_{z+1}$) и $\ln(x/A_z^2)$ (для $x/A_z^2 \in I_{z+}$) ;
либо к паре подзадач $\ln(A_z)$ (для $A_z \in I_{z+1}$) и $\ln(x/A_z)$ (для $x/A_z \in I_{z+}$).

С одной стороны, при таком подходе количество задач удваивается, а с другой стороны, все аргументы вытесняются в I_{z+}. Вытеснение аргументов заводит их в левую окрестность 1, где поведение функции $\ln(x)$ хорошо изучено.

Свойство 6. $\ln(x) \sim x-1$ при $x \to 1$.

Свойство 7. Если $0.5 \le 1-\delta \le x < 1$, то $\quad 0 < x-1-\ln(x) \le \frac{1}{2(1-\delta)}(x-1)^2$.

Не ограничивая общности, будем рассматривать случай $\delta = 2^{-\eta}$, где η – целочисленный параметр точности, $\eta = 2, 3, \ldots$. При этом

Свойство 8. Условия $1-2^{-\eta} \le x < 1, \quad A_\eta \le x < 1, \quad x \in I_{\eta+}$ и $z(x) > \eta$ эквивалентны.

Для аргумента x из $[0.5, 1)$ и положительного параметра точности η рекурсивно определим функцию $\mathrm{Rln}(x;\eta)$ следующим образом:

$$\mathrm{Rln}(x;\eta) \;=\; \begin{cases} x-1, & \text{если } z > \eta, \text{ где } z = z(x), \\ \mathrm{Rln}(x/A_z^2;\eta) + 2\,\mathrm{Rln}(A_z;\eta), & \text{если } z \le \eta \text{ и } x < B_z, \\ \mathrm{Rln}(x/A_z;\eta) + \;\; \mathrm{Rln}(A_z;\eta) - & \text{иначе.} \end{cases} \qquad (4)$$

Идея вычисления $y_0 = \ln(x_0)$ методом вытеснения состоит в том, что $y_0 \approx \mathrm{Rln}(x_0;\eta)$. Параметр η считается известным. Алгоритмическую корректность, т.е. конечность процесса вычисления $\mathrm{Rln}(x_0;\eta)$, гарантируют свойства 5 и 6.

"Врожденным пороком" метода является замена $\ln(x)$ на $x-1$ при $z(x) > \eta$. Общая погрешность вычисления складывается из ошибок округления при выполнении арифметических операций Δ_a и из погрешности замен Δ_e, то есть $|\mathrm{Rln}(x_0;\eta) - y_0| < \Delta_a + \Delta_e$. Для оценки Δ_e определим (из самых общих соображений) верхнюю границу числа замен. Наибольшее количество замен возможно, если:

– исходный аргумент $x_0 \in I_2$, и

– 1 задача для аргумента из I_2 порождает 2 задачи для аргументов из I_3,

– 2 задачи для аргументов из I_3 порождают 4 задачи для аргументов из I_4, и т. д.

В конечном итоге на полуинтервале $I_{\eta+}$ заменами решаются $2^{\eta-1}$ задач. С учетом свойства 7 погрешность Δ_e оценивается так:

$$\Delta_e \;\le\; \frac{1}{2(1-2^{-\eta})}\,(2^{-\eta})^2 \cdot 2^{\eta-1} \;=\; \frac{1}{1-2^{-\eta}}\,2^{-(\eta+2)} \;<\; 2^{-(\eta+1)}.$$

Предложенный в [2] метод редукции аргумента функции $\log_2(x)$ позволет улучшить оценку на два порядка: $\Delta_e \;<\; 2^{-(\eta+3)}$. Вместе с тем, привлечение дополнительного свойства $A_n/A_{n+1}^2 \in I_{2n+2}$ позволяет привести полученную оценку к виду $\Delta_e \;<\; 2^{-(\eta+1)}\,2^{-m(\eta)}$, где $m(\eta)$ – рекуррентно определенная функция поправок, принимающая значения 0, 0, 1, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 7, 7, 8, 9, 9, 10, 11, 11, 12, 13,

13, 14, 15, 16, 16, 17, 18, 18, 19, 20, 21, 22, 22, 23 для $\eta = 2, 3, ..., 38, 40-1$. Так, при $\eta = 15,\ m(\eta) = 6$ и $\Delta_e < 2^{-16}\,2^{-6} = 2^{-22}$.

3. Программирование метода вытеснения. Прежде всего отметим три обстоятельства, важных для практической реализации метода.

1. Задача нормализации чисел требует знания стандартов представления вещественных чисел в компьютере [3]. Не отвлекаясь на эти детали, будем считать известной процедуру FEX(X, P, U), которая по заданному положительному X вычисляет его порядок P и мантиссу U. Результатом FEX(1, P, U) являются P=1 и U=0.5, результатом FEX(5.5, P, U) – P=3 и U=0.6875 и т. д. Кроме того, будем считать фиксированным количество разрядов в двоичном представлении мантиссы. Для определенности будем считать мантиссу сорокаразрядной, полагая, что в настоящем изложении число 40 играет роль параметра.

2. Функция $z(x)$ удобна для записи формул, но необходимость вычисления $z(x)$ определяет переборный характер метода вытеснения. Безусловно, наилучший способ вычисления $z(x)$ состоит в исследовании разрядов двоичного представления x. Фактически $z(x)$ есть номер самого левого нуля в дробной части аргумента: $z(0.625) = z(0.101_2) = 2$, $z(0.828125) = z(0.110101_2) = 3$ и т. д. Однако из методологических соображений для вычисления $z(x)$ будем использовать заранее подготовленный массив-справочник вещественных чисел ABD[1, ... , 40] = $(A_1, ..., A_{40})$. В этом случае $z(\mathrm{X}) = \mathrm{Z}$, если ABD[Z−1] ≤ X < ABD[Z].

Так или иначе, вычисление $z(x)$ сводится к перебору номеров 2, 3, ... до тех пор пока не встретится подходящий номер z, а после его обнаружения возникают две новые задачи $z(A_z)$ и $z(x/A_z^k)$ (для k=1 или k=2), но, как следует из свойства 5, для их решения процесс перебора номеров можно начинать с z+1.

3. Окончание вычислительного процесса определяется условием Z > ETA , где ETA – целочисленная константа. Например, ETA = 15.

Буквальная реализация формулы (4) на языке Паскаль состоит из вспомогательной рекурсивной функции RCR и из основной функции LNR.

```
function RCR(X : real; Z : integer) : integer;
begin   Z:=Z+1;
        if   Z > ETA
        then RCR:=X-1
        else if   X < sqr(ABD[Z])
             then RCR:=RCR(X/sqr(ABD[Z]),Z) + 2*RCR(ABD[Z],Z)
             else if   X < ABD[Z]
                  then RCR:=RCR(X/ABD[Z],Z) +   RCR(ABD[Z],Z)
(* next Z *)       else RCR:=RCR(X,Z)                        end;
```

```
function LNR(X : real) : real;
   var P : integer;
begin   FEX(X,P,X); LNR:=RCR(X,1)-P*RCR(0.5,1)   end;
```

Первая из них выполняет всевозможные проверки и преобразования для аргумента X на полуинтервале I_{Z+1}, а вторая обеспечивает корректные обращения к RCR и вырабатывает искомое приближение ln(X).

Даже при разовом вычислении LNR(X) многократно возникают и решаются "с нуля" совершенно одинаковые задачи вида RCR(ABD[Z], Z). В связи с этим имеет смысл заранее запасти результаты решения этих задач в справочнике ALN[1, .., 40]. Кроме того, для чисел B_Z также имеет смысл завести справочник BBD[1, .., 40], то есть ALN[Z] = RCR(ABD[Z], Z), BBD[Z] = B_Z. В этом случае формула (4) принимает следующий вид:

$$\mathrm{Rln}(x;\eta) = x\prod_{i=2}^{\eta} A_i^{-k_i} - 1 + \sum_{i=2}^{\eta} k_i{\times}\mathrm{ALN}[\,i\,], \qquad \text{где} \quad k_i \in \{0,1,2\}.$$

Соответствующий алгоритм выглядит так:

```
function LNT(X : real) : real;
   var T : real;
       Z : integer;
begin   FEX(X,Z,X);
        T:=-Z*LNA[1];
        for Z:=2 to ETA do
             if X < BBD[Z] then begin X:=X/BBD[Z]; T:=T+2*ALN[Z] end
        else if X < ABD[Z] then begin X:=X/ABD[Z]; T:=T+  ALN[Z] end;
        LNT:=X-1+T                                               end;
```

Временная сложность алгоритма LNT есть $O(\eta)$, а для вычисления первых η значений справочника ALN можно предложить алгоритм с оценкой сложности $O(\eta^2)$. При переходе от LNR к LNT точность вычислений остается неизменной. Заметим, что для фиксированного параметра точности η погрешность вычисления LNT(X) можно уменьшить, если внести в справочник ALN вместо $\mathrm{Rln}(A_z;\eta)$ существенно более точные значения $\ln(A_Z)$, полученные, скажем, иным методом вычисления логарифмов [1, 4].

В приведенном алгоритме источниками погрешности Δ_a являются данные из справочников ALN и BBD, а также операции деления; остальные действия допускают точное выполнение. Общая проблема всех реализаций связана с вычислением значений B_Z. Из-за ограниченности разрядов для записи мантиссы при $Z > 40/2$ числа B_Z теряют младшую единицу и становятся неотличимыми от A_{Z-1}, поэтому условия X < BBD[Z] оказываются ложными, и целая ветвь вычислений оказывается заблокированной. Таким образом, при ETA $> 40/2$ алгоритмы начинают существенно зависеть от

погрешностей округления, что ограничивает их применение. И все-таки ситуация не столь драматична.

4. Обобщение метода. Предложенный в п.2 метод вытеснения можно рассматривать как представителя целого класса методов, отличающихся друг от друга параметрами. Рассмотрим один из способов параметризации, порождающий обобщенный метод вытеснения, схематично представленный на рисунке $3a$. Формально обобщенный метод описывается системой ограничений на значения параметров E_n и R_n:

$$\begin{cases} A_{n-1} \le E_n < A_n, & \text{что соответствует } E_n \in I_n; \\ A_n \le \dfrac{A_{n-1}}{R_n A_n} \le \dfrac{E_n}{R_n A_n} \le 1, & [A_{n-1}, E_n)/(R_n A_n) \subseteq I_{n+}; \\ A_n \le R_n < 1, & R_n \in I_{n+}; \\ A_n \le \dfrac{E_n}{A_n} < 1, & [E_n, A_n)/A_n \subseteq I_{n+}. \end{cases}$$

Эквивалентными преобразованиями эта система сводится к неравенствам:

$$\begin{cases} B_n \le E_n \le C_n \\ E_n/A_n \le R_n \le C_n/A_n \end{cases} \tag{5}$$

Пары (E_n, R_n), удовлетворяющие (5), образуют треугольную область ER_n, изображенную на рисунке $3b$.

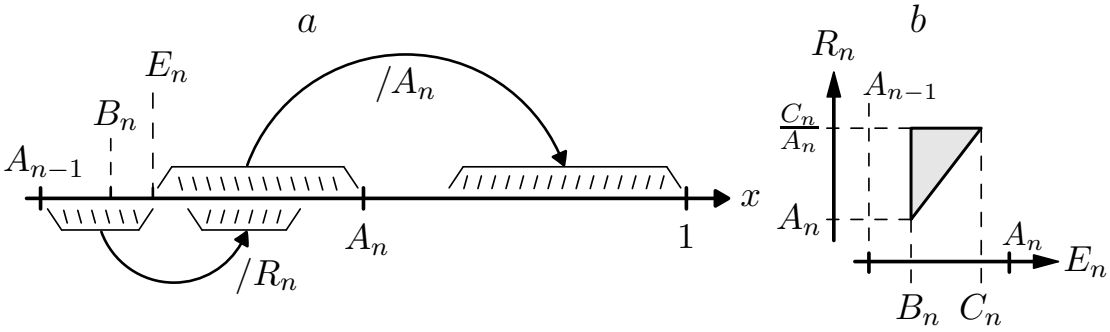

Рис. 3 Схема обобщенного метода и область ER_n

Свойство 9 (обобщение свойства 5). Если $x \in I_z$ и $(E_z, R_z) \in ER_z$, то справедливы формула (6) и соотношения (7):

$$\ln(x) = \begin{cases} \ln((x/R_z)/A_z) + \ln(A_z) + \ln(R_z), & \text{если } x < E_z, \\ \ln(x/A_z) + \ln(A_z), \quad - & \text{иначе;} \end{cases} \tag{6}$$

$$\left.\begin{array}{r} A_z \in I_{z+1} \text{ и } R_z \in I_{z+}, \\ \text{если } x < E_z, \text{ то } (x/R_z)/A_z \in I_{z+}, \\ \text{если } x \ge E_z, \text{ то } \quad x/A_z \in I_{z+} \end{array}\right\}. \tag{7}$$

При фиксированном способе выбора E_n и R_n вычисление логарифма с помощью обобщенного метода вытеснения задается формулой

$$\mathrm{Rlm}(x;\eta) \;=\; \begin{cases} x-1, & \text{если } z=z(x)>\eta; \\ \mathrm{Rlm}((x/R_z)/A_z;\eta)+\mathrm{Rlm}(A_z;\eta)+\mathrm{Rlm}(R_z;\eta), & \text{если } z\le\eta \text{ и } x<E_z; \\ \mathrm{Rlm}(\ x/A_z;\eta)\ +\mathrm{Rlm}(A_z;\eta)\ - & \text{иначе.} \end{cases}$$

При некоторых значениях E_n и R_n обобщенный метод вытеснения приобретает дополнительные свойства, способные повлиять и на область применимости алгоритма, и на выбор справочников. В частности, все ER_n содержат пары $(E_n^{'}, A_{n+1})$, где $E_n^{'} = (A_{n-1}+A_n)/2$ – середина полуинтервала I_n. Использование $E_n^{'}$ и A_{n+1} в качестве значений E_n и R_n порождает алгоритм, который зависит от ошибок округления при ETA $>$ 40 – 2, что вполне приемлемо для реальных задач.

В общем случае области ER_n обладают сеточным строением, как показано на рис. 4 и сформулировано в виде свойства 10.

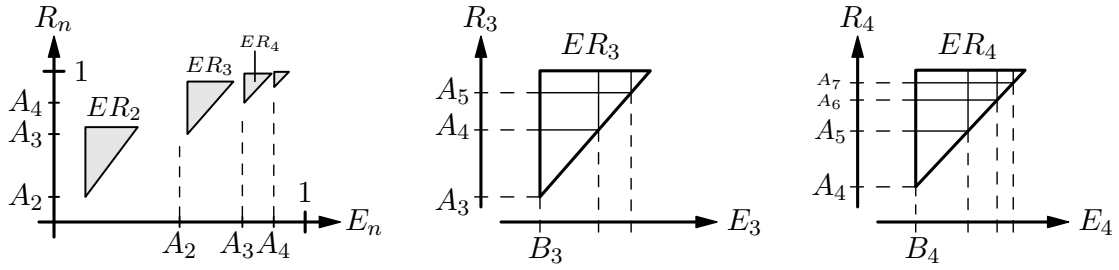

Рис. 4 Строение областей ER_n

Свойство 10. Для любого $n \ge 2$ справедливы неравенства:

$$\begin{array}{rcccccccl} A_n & < & A_{n+1} & < & \ldots & < & A_r & < & C_n/A_n, \ \text{где } r=z(C_n/A_n), \text{ и} \\ B_n = A_nA_n & < & A_{n+1}A_n & < & \ldots & < & A_rA_n & < & C_n. \end{array}$$

Таким образом, узлами сетки для области ER_n являются:
1) пары $(A_kA_n\,,\,A_m)$, где $(k\,,m) \in \{n, n+1, \ldots, r\}^2$ и $k \le m$ *(случай 1)*;
2) пары $(A_jA_n\,,\,C_n/A_n)$, где $j = n, n+1, \ldots, r$ *(случай 2)*;
3) пара $(C_n\,,\,C_n/A_n)$ *(случай 3)* .

Если в формуле (6) на полуинтервале I_z значения E_z и R_z выбраны равными соответственно $A_{z+p}A_z$ и A_{z+q} (случай 1, n=z, k=z+p, m=z+q), то соотношения (7) принимают следующий уточненный вид:

$$\left.\begin{array}{r} A_z \in I_{z+1} \text{ и } R_z \in I_{(z+q)+}, \\ \text{если } E_z \le x, \text{ то} \qquad x/A_z \in I_{(z+p)+}, \\ \text{если } E_z > x, \text{ то } (x/R_z)/A_z \in I_{(z+p)+} \end{array}\right\}.$$

Другими словами, задача вычисления $\ln(x)$ (для $x \in I_z$) сводится к подзадаче $\ln(A_z)$ (для $A_z \in I_{z+1}$), которая решается:
– либо совместно с подзадачей $\ln(x/A_z)$ (для $x/A_z \in I_{(z+p)+}$);
– либо совместно с подзадачами $\ln((x/R_z)/A_z)$ (для $(x/R_z)/A_z \in I_{(z+p)+}$) и $\ln(R_z)$ (для $R_z \in I_{(z+q)+}$).
Только одна из перечисленных подзадач – подзадача вычисления $\ln(A_z)$ – вытесняется в следующий полуинтервал I_{z+1}, в то время как остальные подзадачи вытесняются вперед на p и q полуинтервалов. На первый взгляд, E_n и R_n следует выбирать так, чтобы числа p и q были максимальными, а еще лучше использовать $E_n = C_n$, $R_n = C_n/A_n$. Однако при существенных отклонениях параметров E_n и R_n от медианных значений алгоритм начинает зависеть от погрешностей округления.

5. Заключительные замечания. 1. Почти все операции, задействованные в методе вытеснения, реализуются быстрыми командами логического типа [2]. Единственное исключение составляет операция деления вещественных чисел, но и она допускает определенную свободу в выборе делителя, что позволяет контролировать ошибку округления. 2. Вычислительные эксперименты подтверждают работоспособность и эффективность метода. 3. Метод вытеснения допускает замену степеней двойки на степени чисел из $[2, 3)$. Варьирование основаниями степеней способно сократить объем вычислений, но исключает использование команд логического типа. 4. Метод вытеснения можно успешно применять и для вычисления квадратных корней.

СПИСОК ЛИТЕРАТУРЫ

1. Brent R., Zimmermann P. Modern Computer Arithmetic. Cambridge University Press, 2011.
2. 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.
3. Overton M. L. Numerical Computing with IEEE Floating Point Arithmetic. Philadelphia: SIAM, 2001.
4. Сальников М. С. Рекурсивный алгоритм вычисления логарифма // Информационные процессы. 2012. **12**. № 3. С. 248–252.

Поступила в редакцию
12.09.12