Информационные процессы, Том 18, № 4, 2018, стр. 386–392.

МАТЕМАТИЧЕСКИЕ МОДЕЛИ, ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ

Таблицы точных произведений[1]

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

МГУ имени М.В.Ломоносова, факультет ВМК, Москва, Россия
vlabr@cs.msu.su, nbaeva@gmail.com, soloviev@glossary.ru
Поступила в редколлегию 10.12.2018

Аннотация—В работе предлагаются особые структуры данных, названные таблицами точных произведений, обсуждаются вопросы их применения и ставится задача разработки практически применимых алгоритмов, ориентированных на построение таких таблиц. Исходя из понимания переборного характера упомянутых алгоритмов, в работе сформулированы и доказаны специальные свойства точных произведений, позволяющие отсекать на ранних этапах заведомо неуспешные варианты. Показывается, что учет специальных свойств существенно влияет на структуру переборного алгоритма и, в конечном итоге, позволяет строить практически значимые таблицы точных произведений.

КЛЮЧЕВЫЕ СЛОВА: алгоритм, применимость алгоритма, перебор, сомножители, таблица.

1. ВВЕДЕНИЕ

Прогресс вычислительной техники предоставил в распоряжение программистов фантастические объемы оперативной памяти и породил новый класс задач, так или иначе связанных с порождением и обработкой сверхбольших данных. В настоящей работе рассматривается одна из таких задач, состоящая в формировании специфической таблицы умножения, размеры которой ограничены размерами устоявшихся форматов чисел. По сути дела, речь идет о перечислении всех точных произведений, которые могут возникнуть в ходе вычислений, использующих стандартные типы данных. Предлагаемое исследование точных произведений, сведенных в единую таблицу, состоит из четырех частей:

- в п.2 приводится определение и формулируется задача построения таблиц точных произведений;
- в п.3 формулируются и доказываются свойства таблиц точных произведений, ориентированные на организацию эффективного процесса их построения;
- в п.4. обсуждаются и описываются алгоритмы построения таблиц;
- в заключении обсуждаются вопросы практического применения таблиц точных произведений.

2. ТАБЛИЦЫ ТОЧНЫХ ПРОИЗВЕДЕНИЙ

Пусть n – натуральное число,

- обозначим H_n множество троек натуральных чисел (a, b, c), удовлетворяющих условиям

$$\frac{a}{2^n} = \frac{b}{2^n} \times \frac{c}{2^n}, \qquad \text{где} \quad 2^{n-1} < a, b, c < 2^n, \tag{1}$$

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

- положим $F_n(a) = \{\, b \,|\, (a,b,c) \in H_n \text{ для нек. } c \,\}$ и
- положим $T_n = \{\, (a,b,c) \in H_n \,|\, b = \min F_n(a) \,\}$.

Будем называть: произведением – первый элемент упорядоченной тройки, сомножителями – второй и третий элементы тройки, таблицей точных произведений – множество T_n, размерностью таблицы – число n. Приведем примеры таких таблиц:

$T_2 = \varnothing$, $T_3 = \varnothing$,
$T_4 = \{\, (9,12,12) \,\}$, $T_5 = \{\, (18,24,24), (21,24,28) \,\}$,
$T_6 = \{\, (33,44,48), (35,40,56), (36,48,48), (39,48,52), (42,48,56), (45,48,60), (49,56,56) \,\}$.

С содержательной точки зрения таблица, скажем, T_5 задает два очевидных равенства

$$\frac{18}{32} = \frac{24}{32} \times \frac{24}{32} \qquad \text{и} \qquad \frac{21}{32} = \frac{24}{32} \times \frac{28}{32}.$$

При этом утверждается, что для знаменателя $2^5 = 32$ других равенств, удовлетворяющих условиям (1) не существует.

Нетрудно заметить, что T_n содержит все двукратно увеличенные копии троек из T_{n-1}:

$$\{\, (2a,2b,2c) \,|\, (a,b,c) \in T_{n-1} \,\} \subseteq T_n,$$

поэтому с точки зрения построения таблиц точных произведений интерес представляют их нетривиальные подмножества:

$$T_n^o = T_n \setminus \{\, (2a,2b,2c) \,|\, (a,b,c) \in T_{n-1} \,\}. \tag{2}$$

Приведем некоторые подмножества T_n^o:

$T_2^o = \varnothing$, $T_3^o = \varnothing$,
$T_4^o = \{\, (9,12,12) \,\}$, $T_5^o = \{\, (21,24,28) \,\}$,
$T_6^o = \{\, (33,44,48), (35,40,56), (39,48,52), (45,48,60), (49,56,56) \,\}$.

Подмножества T_n^o позволяют построить любую таблицу точных произведений:

$$\begin{aligned} T_n &= T_n^o \cup \{\, (2a,2b,2c) \,|\, (a,b,c) \in T_{n-1} \,\} = \\ &= T_n^o \cup \{\, (2a,2b,2c) \,|\, (a,b,c) \in T_{n-1}^o \,\} \cup \{\, (4a,4b,4c) \,|\, (a,b,c) \in T_{n-2} \,\} = \cdots \end{aligned}$$

Окончательно: $T_n = \bigcup\limits_{i=4}^{n} \left\{\, (2^{n-i}a,\, 2^{n-i}b,\, 2^{n-i}c) \,|\, (a,b,c) \in T_{n-1}^o \,\right\}$ для $n \geq 4$.

Пусть n – количество разрядов, отведенных под двоичную мантиссу[1] в некотором формате представления вещественных чисел [1]. В этом случае каждая тройка (a,b,c) из T_n задает конкретную операцию умножения, которая выполняется без округлений и отбрасывания разрядов.

3. ФОРМАЛЬНЫЕ СВОЙСТВА ТАБЛИЦ ТОЧНЫХ ПРОИЗВЕДЕНИЙ

Все дроби, фигурирующие в условиях (1), принадлежат интервалу (0.5; 1). Это обстоятельство порождает

Свойство 1. Для любой тройки (a,b,c) из T_n имеют место неравенства $a < b$ и $a < c$.

[1] При этом считается, что старшая, “невидимая” единица мантиссы занимает один из n разрядов.

Учитывая свойство 1 и определение таблиц T_n, преобразуем условия (1) к следующему эквивалентному виду:

$$a\,2^n = b \cdot c, \qquad \text{где} \quad 2^{n-1} < a < b \le c < 2^n. \tag{3}$$

Свойство 2. В любой тройке (a, b, c) из T_n^o, $(n \ge 4)$ число a является нечетным.

Доказательство. Пусть для тройки (a', b', c') из T_n^o число a' является четным. Предположим, что b' – нечетное. В этом случае в разложении на простые множители числа c' имеются по меньшей мере $n{+}1$ двоек, то есть $c' \ge 2^{n+1}$, что противоречит условиям (1). Итак, в тройке (a', b', c') из T_n^o все числа являются четными, и, учитывая (2), $(a', b', c') \notin T_n^o$. Свойство доказано.

Любое натуральное число d можно однозначно представить в виде

$$d = d_1\,2^{d_2}, \qquad \text{где} \quad d_1, d_2 - \text{натуральные, причем} \;\; d_1 - \text{нечетное и} \;\; d_2 \ge 0. \tag{4}$$

Будем рассматривать представления (4) для множителей b и c из (3). $b = b_1\,2^{b_2}$, $c = c_1\,2^{c_2}$.

Свойство 3. Для любой тройки (a, b, c) из T_n^o, $(n \ge 4)$ выполняются соотношения:

$$3 \;\le\; b_1, \quad 3 \;\le\; c_1, \qquad 2 \;\le\; b_2,\, c_2 \;\le\; n{-}2, \qquad b_2 + c_2 = n.$$

Доказательство. Пусть (a, b, c) – произвольная тройка из T_n^o. $(3) \Rightarrow a\,2^n = b{\cdot}c = b_1\,c_1\,2^{b_2+c_2}$. Поскольку a, b_1 и c_1 нечетные, то $n = b_2 + c_2$, то есть формально b_2 и c_2 могут принимать значения $0, 1, \ldots, n$. Если предположить, что $b_2 = n$, то $b = b_1\,2^n \;\Rightarrow\; b \ge 2^n$, что противоречит условиям (1). Так как $b_2 \ne n$, то $c_2 \ne 0$. Аналогично, $c_2 \ne n$ и $b_2 \ne 0$.

Итак, $1 \le b_2, c_2 \le n{-}1$. Если предположить, что $b_1 = 1$, то $b = 2^{b_2} \;\Rightarrow\; b \le 2^{n-1}$, что противоречит условиям (1). $b_1 \ne 1 \;\Rightarrow\; b_1 \ge 3$. Аналогично, $c_1 \ge 3$.

Если предположить, что $b_2 = n{-}1$, то $b = b_1\,2^{n-1} = (3{+}\delta)\,2^{n-1} \;\Rightarrow\; b > 2^n$, что противоречит условиям (1). Так как $b_2 \ne n{-}1$, то $c_2 \ne 1$. Аналогично, $c_2 \ne n{-}1$ и $b_2 \ne 1$. Свойство доказано.

Следствие. Если (a', b', c') – тройка из T_n^o, то числа b' и c' кратны четырем: $b' = 4\,b''$, $c' = 4\,c''$ для некоторых натуральных b'' и c''.

Свойство 4. Для любой тройки (a, b, c) из T_n^o $(n \ge 4)$ выполняются соотношения:

$$a = b_1 \cdot c_1, \qquad 2^{c_2-1} \;<\; b_1 \;<\; 2^{c_2}, \qquad 2^{n-c_2-1} \;<\; c_1 \;<\; 2^{n-c_2} \qquad \text{для} \quad c_2 = 2, \ldots, n{-}2.$$

Доказательство. $a\,2^n = b \cdot c = b_1\,2^{b_2} \cdot c_1\,2^{c_2} \;\Rightarrow\; a = b_1 \cdot c_1$.
$2^{n-1} < c < 2^n \;\Rightarrow\; 2^{n-1} < c_1\,2^{c_2} < 2^n \;\Rightarrow\; 2^{n-c_2-1} < c_1 < 2^{n-c_2}$.
$2^{n-1} < b < 2^n \;\Rightarrow\; 2^{n-1} < b_1\,2^{n-c_2} < 2^n \;\Rightarrow\; 2^{c_2-1} < b_1 < 2^{c_2}$. Свойство доказано.

Для множества нечетных чисел из интервала $(d_1; d_2)$ будем использовать обозначение $\langle d_1; d_2\rangle$. Например, $\langle 2^{2-1}; 2^2\rangle = \{3\}$, $\langle 2^{4-1}; 2^4\rangle = \{9, 11, 13, 15\}$, $\langle 33; 51\rangle = \{35, 37, 39, 41, 43, 45, 47, 49\}$. Поскольку b_1 и c_1 нечетные числа, то свойство 4 можно переформулировать в следующем виде:

Свойство 4'. Для любой тройки (a, b, c) из T_n^o $(n \ge 4)$ выполняются соотношения:

$$a = b_1 \cdot c_1, \qquad b_1 \in \langle 2^{c_2-1}; 2^{c_2}\rangle, \qquad c_1 \in \langle 2^{n-c_2-1}; 2^{n-c_2}\rangle \qquad \text{для} \quad c_2 = 2, \ldots, n{-}2.$$

Для произвольной пары чисел b_1 и c_1 из $\langle 2^{c_2-1}; 2^{c2}\rangle$ и $\langle 2^{n-c_2-1}; 2^{n-c2}\rangle$ неравенство $2^{n-1} < b_1 \cdot c_1 < 2^n$ может не выполняться. Если, например, $b_1' = 2^{c_2-1} + 1$, $c_1' = 2^{n-c_2-1} + 1$ и $c_2 = 3$, то $b_1' \cdot c_1' < 2^{n-1}$. Вместе с тем, $2^{n-1} < (2^{c_2}{-}1)\,(2^{n-c_2}{-}1) < 2^n$ для любого $c_2{=}2, \ldots, n{-}2$.

Введем обозначение для наибольшего произведения из T_n^o: $a_{\max}(n) = \max\{\, a \,|\, (a, b, c) \in T_n^o \,\}$.

Свойство 5. $a_{\max}(n) = \begin{cases} 2^n + 1 - 3 \cdot 2^k, & \text{если } n = 2k+1; \\ 2^n + 1 - 2 \cdot 2^k, & \text{если } n = 2k. \end{cases}$

Доказательство. Как следует из свойств 3 и 4
$a_{\max}(n) = \max\{\,(2^{c_2}-1)\,(2^{n-c_2}-1) \,|\, c_2{=}2,\ldots,n{-}2\,\} = 2^n + 1 - \min\{\,2^i{+}2^{n-i} \,|\, i{=}2,\ldots,n{-}2\,\}$.
Если $n = 2k+1$, то элементы множества $\{\,2^i{+}2^{n-i} \,|\, i{=}2,\ldots,n{-}2\,\}$ упорядочены так:

$$2^2{+}2^{n-2} > \cdots > 2^{k-1}{+}2^{k+2} > 2^k{+}2^{k+1} \;=\; 2^{k+1}{+}2^k < 2^{k+2}{+}2^{k-1} < \cdots < 2^{n-2}{+}2^2.$$

Если $n = 2k$, то

$$2^2{+}2^{n-2} > \cdots > 2^{k-1}{+}2^{k+1} \;>\; 2^k{+}2^k \;<\; 2^{k+1}{+}2^{k-1} < 2^{k+2}{+}2^{k-2} < \cdots < 2^{n-2}{+}2^2.$$

То есть $a_{\max}(2k{+}1) = 2^n + 1 - (2^k{+}2^{k+1})$ и $a_{\max}(2k) = 2^n + 1 - (2^k{+}2^k)$. Свойство доказано.

4. АЛГОРИТМЫ ПОСТРОЕНИЯ ТАБЛИЦ ТОЧНЫХ ПРОИЗВЕДЕНИЙ

Задача построения таблиц точных произведений решается перебором троек. Известно, что применимость переборных алгоритмов в конечном итоге ограничивается наличными вычислительными ресурсами. Вместе с тем, несколько раздвинуть рамки практической применимости переборных алгоритмов можно за счет привлечения к организации перебора тонких свойств исходной задачи. Приведем два алгоритма построения таблиц точных произведений. Первый из них реализует простой перебор, минимально использующий особенности задачи. Второй – использует ряд дополнительных особенностей, ускоряющих вычисления, но снижающих понятность алгоритма.

Для записи алгоритмов будем использовать псевдокод, восходящий к [2]. Несущественные технические детали вычислений будем считать реализованными в алгоритмах `pwr`, `out` и `sup`:

- алгоритм `pwr(d,d1,d2)` по заданному `d` находит числа `d1` и `d2`, удовлетворяющие (4);
- алгоритм `out(a,b,c)` добавляет в итоговую таблицу точных произведений новую тройку `(a,b,c)`;
- алгоритм `sup(a)` вычисляет верхнюю границу для меньшего из двух сомножителей, произведение которых есть `a`: если `a = x y` и `x ≤ y`, то `x ≤ sup(a)`.

Первый алгоритм `TripletsSimple` – см. Алгоритм 1 – предназначен для построения множества T_n^o по заданной размерности $n \geq 4$. Алгоритм `TripletsSimple` фиксирует основную идею перебора потенциально возможных произведений.

Алгоритм 1. Простой алгоритм построения таблицы T_n^o.

```
Algorithm TripletsSimple(n)
// n - натуральное; n ≥ 4
{  for a:=2^(n-1) + 1 to a_max(n) step 2 do {
       for b:=a + 4 - (a and 3) to 2^n-1 step 4 do {
           pwr(b,b1,b2);
           if a mod b1 = 0 then {
               c:=(a div b1)*2^(n-b2);
               if c ∈ (2^(n-1); 2^n) then out(a,b,c);
}  }    }  }
```

Комментарий 1. Первый оператор цикла перебирает всех кандидатов в произведения с целью их последующей проверки. При этом существенно используется свойство 2.

Комментарий 2. Второй оператор цикла перебирает для заданного числа `a` все возможные значения множителя `b`. При этом используются свойство 1 и следствие из свойства 3. Выражение $a + 4 - (a \text{ and } 3)$ вычисляет ближайшее целое, кратное 4 и превосходящее `a`.

Комментарий 3. Выражение (a div b1)*2^{n-b2}, предназначенное для вычисления второго множителя `c`, основывается на свойстве 3.

Алгоритм `TripletsSimple` имеет смысл использовать для построения таблиц небольших размерностей. Его главное предназначение состоит в построении отладочных тестов для проверки более изощренных алгоритмов.

Второй алгоритм `TripletsModified` – см. Алгоритм 2 – также предназначен для построения множества T_n^o по заданной размерности $n \geq 4$. Алгоритм `TripletsModified` опирается на свойство 4, которое устанавливает связь – см. Таблицу 1 – между точными произведениями и специальными разложениями числа a на сомножители b_1 и c_1.

Таблица 1. Специальные разложения на множители.

c_2	$a = b_1\, c_1$			
2	$b_1 \in \langle 2^1;\, 2^2\rangle$	$\times 2^{n-2}$	$c_1 \in \langle 2^{n-3};\, 2^{n-2}\rangle$	$\times 2^2$
3	$b_1 \in \langle 2^2;\, 2^3\rangle$	$\times 2^{n-3}$	$c_1 \in \langle 2^{n-4};\, 2^{n-3}\rangle$	$\times 2^3$
$\dots$	$\dots$		$\dots$	
$n-3$	$b_1 \in \langle 2^{n-4};\, 2^{n-3}\rangle$	$\times 2^3$	$c_1 \in \langle 2^2;\, 2^3\rangle$	$\times 2^{n-3}$
$n-2$	$b_1 \in \langle 2^{n-3};\, 2^{n-2}\rangle$	$\times 2^2$	$c_1 \in \langle 2^1;\, 2^2\rangle$	$\times 2^{n-2}$
	b		c	

В этих разложениях специальные ограничения, которым должны удовлетворять сомножители, зависят от параметра c_2. Кроме того, специальные ограничения гарантирует взаимозаменяемость сомножителей b_1' и c_1', а значит для нахождения специальных сомножителей достаточно использовать часть ограничений – см. Таблицы 2 и 3.

Таблица 2. Рабочий набор специальных разложений для $n = 2\,k{+}1$.

c_2	$a = b_1\, c_1$			
2	$b_1 \in \langle 2^1;\, 2^2\rangle$	$\times 2^{n-2}$	$c_1 \in \langle 2^{n-3};\, 2^{n-2}\rangle$	$\times 2^2$
3	$b_1 \in \langle 2^2;\, 2^3\rangle$	$\times 2^{n-3}$	$c_1 \in \langle 2^{n-4};\, 2^{n-3}\rangle$	$\times 2^3$
$\dots$	$\dots$		$\dots$	
$k-1$	$b_1 \in \langle 2^{k-2};\, 2^{k-1}\rangle$	$\times 2^{k+2}$	$c_1 \in \langle 2^{k+1};\, 2^{k+2}\rangle$	$\times 2^{k-1}$
k	$b_1 \in \langle 2^{k-1};\, 2^k\rangle$	$\times 2^{k+1}$	$c_1 \in \langle 2^k;\, 2^{k+1}\rangle$	$\times 2^k$
	b		c	

Нетрудно заметить, что при нечетных n в специальных разложениях $a = b_1 \cdot c_1$ сомножители b_1 и c_1 всегда принадлежат разным областям, а при четных $n = 2k$ – см. Таблицу 3 – сомножители b_1 и c_1 при $c_2 = k$ принадлежат одному и тому же множеству $\langle 2^{k-1};\, 2^k\rangle$. Последнее обстоятельство порождает подзадачу выявления и отбраковывания повторно построенных пар сомножителей. С этой целью предлагается использовать верхнюю границу для наименьшего сомножителя – число `sup(a)`.

Заметим, что при последовательном переборе значений $a = a_0$, $a = a_0 + 2$, $a = a_0 + 4$ и т.д. переход от известного значения `sup(`$a_0{+}2i{-}2$`)` к `sup(`$a_0{+}2i$`)` реализуется (в худшем случае) пятью операциями сложения и двумя операциями сравнения. Алгоритм `TripletsModified` исходит из того, что для заданного кандидата `a`, вообще говоря, найдутся несколько пар специальных сомножителей `b1` и `c1`, и поэтому параллельно с подбором пар реализуется (посредством переменных `d1` и `d2`) выбор пары с минимальным значением `b`. Каждая вновь найденная пара

Таблица 3. Рабочий набор специальных разложений для $n = 2\,k$.

c_2	$a = b_1\,c_1$			
2	$b_1 \in \langle 2^1;\, 2^2\rangle$	$\times 2^{n-2}$	$c_1 \in \langle 2^{n-3};\, 2^{n-2}\rangle$	$\times 2^2$
3	$b_1 \in \langle 2^2;\, 2^3\rangle$	$\times 2^{n-3}$	$c_1 \in \langle 2^{n-4};\, 2^{n-3}\rangle$	$\times 2^3$
$\dots$	$\dots$		$\dots$	
$k-1$	$b_1 \in \langle 2^{k-2};\, 2^{k-1}\rangle$	$\times 2^{k+1}$	$c_1 \in \langle 2^k;\, 2^{k+1}\rangle$	$\times 2^{k-1}$
k	$b_1 \in \langle 2^{k-1};\, 2^k\rangle$	$\times 2^k$	$c_1 \in \langle 2^{k-1};\, 2^k\rangle$	$\times 2^k$
	b		c	

`b1` и `c1` изменяет верхнюю границу `s` значений `b1` таким образом, что для всех последующих пар $\texttt{b1}'$ и $\texttt{c1}'$ гарантируется неравенство $\texttt{b1}' < \texttt{b1}$.

Алгоритм 2. Модифицированный алгоритм построения таблицы T_n^o.

```
Algorithm TripletsModified(n)
// n - натуральное; n ≥ 4
{  for a:=2^(n-1) + 1 to a_max(n) step 2 do {
       s:=sup(a)/2^(n-1);
       d1:=-1;
       for c2:=2 to n div 2 do {
          s:=2*s;
          for b1:=2^(c2-1) + 1 to min(⌊s⌋, 2^c2-1) step 2 do
          if  a mod b1  = 0                then
          if (a div b1) ∈ (2^(n-c2-1); 2^(n-c2)) then {   // N.B.
              s:=b1;
              d1:=b1; d2:=n-c2;   // save b
              break;              // next c2
       }  };
       if 0 < d1 then {
          b:=d1*2^d2;
          c:=(a div d1)*2^(n-d2);
          out(a,b,c);
}  }    }
```

Комментарий 4. В алгоритме `TripletsModified` верхняя граница `s` для сомножителей `b1` может принимать не только целочисленные значения, в связи с этим в выражениях с целочисленными операндами используется функция $\lfloor\cdot\rfloor$ отбрасывания дробной части.

Комментарий 5. Если для $\texttt{b1} \in \langle 2^{c2-1};\, 2^{c2}\rangle$ пара `b1` и `c1` является специальным разложением, то дальнейшее исследование множества $\langle 2^{c2-1};\, 2^{c2}\rangle$ следует прекратить. Это соображение реализует оператор `break`.

С точки зрения практической применимости алгоритм `TripletsModified` расширяет возможности алгоритма `TripletsSimple` и позволяет сформировать на современных вычислительных установках таблицы точных произведений, размерность которых соответствует количеству разрядов в мантиссах общепринятых [1] форматов вещественных чисел: Half, Single, Double.

5. ЗАКЛЮЧЕНИЕ

С теоретической точки зрения остается открытым вопрос о существовании для одного и того же произведения a двух различных пар множителей b, c и b', c':

$$\frac{a}{2^n} = \frac{b}{2^n} \times \frac{c}{2^n} = \frac{b'}{2^n} \times \frac{c'}{2^n}, \qquad b \neq b', \qquad b \neq c'.$$

С практической точки зрения таблицы точных произведений представляют интерес для отладки и сравнения вычислительных алгоритмов, а также для конструирования новых аппроксимаций, основанных на тотальных [3] таблицах значений. В части сравнения и отладки таблица позволяет проверить формальные свойства показательных, степенных и логарифмических функций. Скажем, показатель

$$\Phi[\widehat{ln}] = \sum_{(a,b,c)\in T_n} \left| \widehat{ln}\Big(\frac{a}{2^n}\Big) - \widehat{ln}\Big(\frac{b}{2^n}\Big) - \widehat{ln}\Big(\frac{c}{2^n}\Big) \right|$$

можно использовать для сравнения аппроксимаций $\widehat{ln}$ функции вычисления логарифмов: из двух аппроксимаций предпочтительнее аппроксимация с меньшим показателем Φ.

СПИСОК ЛИТЕРАТУРЫ

1. *IEEE Standard for Floating-Point Arithmetic 754-2008*. IEEE, 2008.
2. Horowitz E., Sahni S., Rajasekaran S. *Computer Algorithms*. Computer Science Press, NY, 1998.
3. Абрамов В.Г., Баева Н.В., Казаков А.А., Соловьев С.Ю. Таблично ориентированный подход к нахождению значений функций одной вещественной переменной *Int. J. of Open Information Technologies* 2017, том 5, No. 1 стр. 88-91.

Tables of Exact Products

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

In this paper we (a) describe a special data structures called tables of exact products, (b) discuss their application, and (c) set the task of developing practically applicable algorithms for this tables creating. Since the algorithm has a search character, we formulated and proved special properties of exact products, which allow to cut off at an early stage unsuccessful variants. Also we show that special properties have an impact on the structure of the search algorithm and they allowe to build practically significant tables of exact products.

KEYWORDS: algorithm, applicability of the algorithm, search, multipliers, table.