ОДНОРОДНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ
(Вычислительные системы)
1981 год Выпуск 90

УДК 681.323

ТАБЛИЧНЫЕ ПРОЦЕССОРЫ С НЕРАВНОМЕРНЫМИ ИНТЕРВАЛАМИ

Л.И.Кужий, Б.А.Попов, Я.И.Фет

I. В в е д е н и е. При решении различных научно-технических задач вычисление элементарных и специальных функций занимает значительную долю машинного времени. В некоторых случаях эта доля доходит до 50% [I,2]. Как правило, эти вычисления выполняются на ЭВМ с помощью стандартных программ, построенных на основе наилучших равномерных приближений многочленами или рациональными выражениями [3]. Но такие вычисления часто оказываются неудовлетворительными по быстродействию. Так, для вычисления $tg\frac{\pi}{4}x$ на промежутке $[0,1]$ с точностью 10^{-6} необходимо вычислить полином $P_{2m}(x)$ при $m = 6$, на что требуется I4 машинных операций, или рациональное выражение с пятью параметрами, на что требуется 9 операций (в том числе операция деления) (см. [3]).

Значительно ускорить вычисление функций можно с помощью таблично-интерполяционных методов [4-7]. Обычно используется равномерное деление интервала приближения, при котором ошибки на разных подынтервалах могут существенно различаться между собой. Так как максимальная ошибка на всем интервале приближения определяется наибольшей из ошибок на подынтервалах, то для ее уменьшения требуется существенно увеличивать количество подынтервалов. В результате для достижения заданной точности недопустимо возрастает объем памяти.

Уменьшения объема памяти можно достигнуть за счет выравнивания величин максимальных ошибок на подынтервалах [8]. При этом интервал приближения оказывается разбитым на неравные подынтервалы. Практическое использование таких "неравномерных" таблиц затруднительно, поскольку в обычных запоминающих устройствах поиск подын-

тервала, соответствующего конкретному значению аргумента, является довольно сложной операцией (см., например, [9]). С другой стороны, известно [10], что такой поиск выполняется достаточно просто в запоминающих устройствах ассоциативного типа, реализующих базовую операцию "поиск ближайшего числа".

В настоящей работе предлагается методика уменьшения памяти, необходимой для хранения таблиц при вычислении функций, основанная на использовании неравномерного разбиения интервала приближения (при постоянной максимальной погрешности) и применении ассоциативного поиска.

В работе показано, что предлагаемая методика позволяет для элементарных функций при заданной точности вычислений и фиксированном быстродействии получить экономию памяти в 2-4 раза, либо при заданном объеме памяти - увеличение точности на порядок.

2. Способ построения приближающих выражений. Для построения кусочно-многочленных приближений для функции $f(x)$ на промежутке $[\alpha,\beta]$ с постоянной максимальной ошибкой на подынтервалах необходимо найти сплайн $S_{m,m}(x) \equiv S_m(x)$ порядка m дефекта m с узлами $z = \{z_i\}_{i=1}^{p+1}$ ($z_1 = \alpha$, $z_{p+1} = \beta$):

$$S_m(x) = \sum_{j=0}^{m} a_{ji} x^j \quad \text{при} \quad x \in [z_i, z_{i+1}],\ i = \overline{1,p},$$

$$S_m(x) \in C[\alpha,\beta].$$

Параметры $A = \{A_i\}_{i=1}^{p}$ ($A_i = \{a_{ji}\}_{j=0}^{m}$) сплайна $S_m(x) = S_m(A, z_i, x)$ и его узлы z необходимо выбрать из условия получения минимальной абсолютной ошибки при заданном количестве звеньев p:

$$\max_{x\in[\alpha,\beta]} |f(x) - S_m(A, z_i, x)| = \min_{A\subset B, z\in U} \max_{x\in[\alpha,\beta]} |f(x) - S_m(B, U_j, x)|,$$

где $U = \{u_i\}_{i=1}^{p+1}$ - множество всевозможных чисел вида $\alpha = u_1 < u_2 < u_3 < \ldots < u_{p+1} = \beta$, а B - открытое множество параметров $\{b_{ij}\}$, для которых удовлетворяются условия

$$\sum_{j=0}^{m} b_{ji} u_i^j = \sum_{j=0}^{m} b_{j,i+1} u_{i+1}^j, \qquad i = \overline{2,p}.$$

В работе [8] показано, что решением поставленной задачи является равномерный аппроксимирующий сплайн с заданным количеством

звеньев, максимальная ошибка приближения ε_0 которого достигается на каждом звене $[z_i, z_{i+1}]$, $i = \overline{1,p}$. Равномерный аппроксимирующий сплайн построен так, что первое его звено является наилучшим равномерным приближением многочленом степени m при $x \in [z_1, z_2]$, а все последующие звенья - наилучшими равномерными приближениями с закрепленными левыми границами.

В основе алгоритма для построения равномерного аппроксимирующего сплайна с заданным количеством звеньев лежит алгоритм пост - роения равномерного аппроксимирующего сплайна с заданной ошибкой ε и минимальным количеством звеньев q для функции $f(x)$, заданной в n точках промежутка $[\alpha,\beta]$:

$$X: \alpha = x_1 < x_2 < \ldots < x_n = \beta \qquad (n >> (m+1)q) .$$

Обозначим через $S_m^{(i)}(x) \equiv \sum_{j=0}^{m} a_{ji} x^j$ i-е звено равномерного аппроксимирующего сплайна с заданной ошибкой и через X_i множе - ство $X_i = X \cap [z_i, z_{i+1}]$:

$$X_i: \quad z_i = x_1^{(i)} < x_2^{(i)} < \ldots < x_{p_i}^{(i)} = z_{i+1}; \qquad X = \bigcup_{i=1}^{p} X_i .$$

Тогда сплайн $S_m(x)$ удовлетворяет следующим условиям:

а) многочлен $S_m^{(1)}(x) = \sum_{j=0}^{m} a_{j1} x^j$ есть наилучшее равномерное приближение функции $f(x)$, заданной в точках X_1;

б) $$\max_{1 \le j \le p_1} |f(x_j) - S_m^{(1)}(x_j)| = \varepsilon_1 \le \varepsilon_0 ;$$

в) $$\max_{1 \le j \le p_1+1} |f(x_j) - S_m^{*}(x_j)| > \varepsilon_0 ,$$

где $S_m^{*}(x)$ - наилучшее равномерное приближение функции $f(x)$ многочленом степени m в точках $X_1^{*} = X_1 \cup X_{p_1+1}$;

г) для $i = \overline{2,p}$ многочлен $S_m^{(i)}(x)$ есть наилучшее равномерное приближение функции $f(x)$, заданной в точках X_i, с условием

$$S_m^{(i-1)}(x_i) = S_m^{(i)}(z_i) ; \qquad (1)$$

д) $\max\limits_{2\le j\le p_i} |f(x_j^{(i)}) - S_m^{(i)}(x_j^{(i)})| = \varepsilon_i \le \varepsilon_0 , \quad i=\overline{2,p}$;

е) $\max\limits_{2\le j\le p_i+1} |f(x_j^{(i)}) - S_m^{*}(x_j^{(i)})| > \varepsilon_0 , \quad i=\overline{2,p-1}$,

где $S_m^{*}(x)$ - наилучшее равномерное приближение функции $f(x)$ с условием (I) многочленом степени m в точках $X_i^{*} = X_i \cup \{X_{p_i+1}\}$.

Машинный алгоритм построения такого сплайна подробно описан в работе [8].

Для построения равномерного аппроксимирующего сплайна $S_m(x)$ с заданным количеством звеньев p для непрерывной функции $f(x)$ на промежутке $[\alpha,\beta]$ выбираем разбиение этого промежутка точками $X_i = \alpha + (i-1)(\beta-\alpha)/(n-1)$, $i=\overline{1,n}$. При достаточно густой сетке $X = \{X_i\}_{i=1}^{n}$ приближения на множествах X и $[\alpha,\beta]$ практически неразличимы [8]. Далее необходимо выбрать начальное значение ошибки приближения ε и построить равномерный аппроксимирующий сплайн с заданной ошибкой $S_m(\varepsilon,x)$. Полученное количество звеньев сплайна обозначаем через q.

При $q>p$ ошибку приближения ε увеличиваем, при $q<p$ - уменьшаем. При $q=p$ ошибку приближения уменьшаем до тех пор, пока незначительное ее уменьшение ведет к увеличению количества звеньев или максимальная ошибка достигается на последнем звене сплайна. В таком случае принимаем, что $S_m(\varepsilon,x) \equiv S_m(x)$.

Результатом вычислений является набор $(m+1)p$ параметров сплайна $S_m(x)$ и его $(p-1)$ узел. В качестве примера в табл.I приведены параметры равномерного параболического сплайн-приближения $S_i(x) = A_i x^2 + B_i x + C_i$, $i=\overline{1,p}$, с минимально возможной ошибкой $\varepsilon = 5{,}9\cdot 10^{-6}$ при $p = 16$ для функции $y = tg\frac{\pi}{4}x$, $0 \le x \le 1$.

3. Техническая реализация процессора. Процессор для вычисления функций таблично-интерполяционным методом должен, вообще говоря, содержать два основных блока: память таблиц для хранения интерполяционных коэффициентов и арифметическое устройство (АУ) для выполнения всех необходимых вычислений (приведение к стандартному интервалу приближения, интерполяция, вычисление искомой функции).

Особенностью рассматриваемого процессора, в соответствии с описанным выше подходом, является неравномерное разбиение стандартного интервала приближения на подынтервалы.

Таблица I

	A_i	B_i	C_i	z_i
I	0,02550435	0,7843927	0,000005870039	0,I049
2	0,06478856	0,777I363	0,0003347808	0,I835
3	0,I053067	0,7632755	0,00I5I39II	0,260I
4	0,I479769	0,742II32	0,004I3I509	0,3344
5	0,I935930	0,7I26736	0,008875II9	0,4060
6	0,2429900	0,6736719	0,0I656742	0,4747
7	0,2968481	0,6237008	0,02815233	0,5402
8	0,3561704	0,5608212	0,04480870	0,6027
9	0,4215957	0,4832361	0,06780367	0,6620
I0	0,4942378	0,3883980	0,09875151	0,7184
II	0,5747749	0,2741003	0,I392979	0,7718
I2	0,6644348	0,I37I918	0,I915557	0,8225
I3	0,7640435	0,02508756	0,2576446	0,8705
I4	0,8750768	0,2I67404	0,34003407	0,9I6I
I5	0,9984532	0,440403	0,4422795	0,9593
I6	I,I346026	0,7003841	0,5657756	I

При равномерном разбиении поиск в памяти таблиц не вызывает затруднений. Обычно применяют следующий простой прием [II]. Аргумент x (n-разрядное двоичное число) разделяется на две части. Старшие N разрядов рассматриваются как значение аргумента в узловой точке x_0 и используются для формирования адреса соответствующих интерполяционных коэффициентов в памяти таблиц (в этом случае в ней хранятся коэффициенты для 2^N равномерно распределенных узлов). Младшие n-N разрядов представляют собой приращение аргумента на данном подынтервале.

При неравномерном разбиении, если использовать обычные запоминающие устройства с адресным обращением, поиск подынтервала становится довольно сложной задачей. Для ускорения поиска в этом случае можно предложить структуру процессора, приведенную на рис.I. Здесь память таблиц содержит два запоминающих устройства: память узловых значений аргумента и память интерполяционных коэффициентов. Аргумент x , поступающий на регистр опроса, сравнивается со

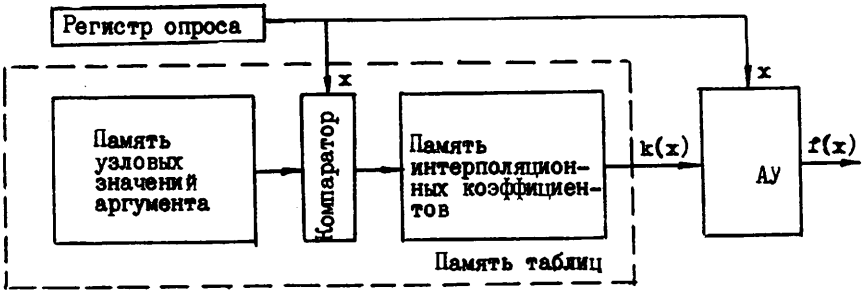

Рис. I

всеми узловыми значениями с помощью параллельного компаратора, который выделяет единственный подынтервал, соответствующий данному x. Выходной сигнал компаратора используется для выборки коэффициентов из памяти коэффициентов.

Главной особенностью данной структуры является необходимость применения операции "поиск ближайшего". Рассмотрим возможности аппаратной реализации этой операции.

Поиск ближайшего - один из видов так называемого сложного ассоциативного поиска. Для выполнения этой операции были предложены различные специализированные устройства [I2,I3]. В общем случае в память такого устройства записываются в произвольном порядке элементы массива аргументов (переменные), а на регистр опроса подается некоторый эталон (константа). Задача заключается в том, чтобы выделить элемент, ближайший к эталону.

В нашем случае можно применить любое из упомянутых специализированных устройств. Однако рассматриваемый табличный процессор можно упростить, если воспользоваться тем обстоятельством, что узловые значения аргументов для каждой функции при выбранной аппроксимации являются по существу константами, вычисляются заранее и могут быть записаны в упорядоченном виде (см., например, табл. I). При этом операция "поиск ближайшего" может быть сведена к операции "поиск бóльших" (или "поиск меньших"), которая реализуется значительно проще. Действительно, пусть в памяти аргументов записаны упорядоченные, например, по возрастанию константы $z_1, z_2, \ldots$ $\ldots, z_p$, а на регистр опроса поступает аргумент x. Предположим, что $z_i \le x < z_{i+1}$. Если компаратор, реализующий операцию "поиск бóльших", отмечает "бóльшие" некоторым сигналом $v = 1$, то строки

памяти аргументов, содержащие $z_1, z_2, \ldots, z_i$, будут отмечены сигналами $v = 0$, а строки, содержащие $z_{i+1}, \ldots, z_p$,- сигналами $v = 1$. Тогда с помощью линейки дополнительных логических схем $w = \bar{v}_i \cdot v_{i+1}$ можно однозначно выделить необходимый подынтервал. Для рассмотренного выше примера $w = 1$ в единственной строке, содержащей z_i. Значит, z_i - ближайшее (меньшее) узловое значение аргумента.

В качестве специализированного устройства, реализующего операцию "поиск больших", можно использовать ε-процессор (ε-матрицу), описанный в [10]. Напомним, что в статическом исполнении ε-процессор представляет собой двумерную однородную логическую сеть, каждая ячейка которой содержит один двоичный запоминающий элемент и несколько вентилей, реализующих функции $z' = z(a \vee \bar{x})$ и $v' = v \vee za\bar{x}$, где a - значение двоичной переменной в запоминающем элементе данной ячейки, а x - значение соответствующего разряда эталона. Как показано в [10], в ε-матрице граничные сигналы $v' = 1$ вырабатываются во всех строках, содержащих числа, большие, чем эталон.

С практической точки зрения целесообразно применить так называемый "динамический" вариант ε-процессора [10]. В этом случае память аргументов выполняется на основе ЗУ с вертикальным обращением, а в компараторе для каждого слова этого ЗУ организуется одна ε-ячейка с соответствующими запоминающими элементами для хранения текущих значений z и v ("триггерами переносов").

4. Сравнение нового метода с ранее известными. Проведем сравнение для части элементарных функций, рассматривавшихся в работах [3-7]: $\sin\frac{\pi}{2}x$, $\log_2(1+x)$, $\mathrm{tg}\frac{\pi}{4}x$, 2^x при $x \in [0,1]$. Если указанные функции заданы в более широкой области, то они сводятся к промежутку $[0,1]$ с помощью стандартных формул приведения [14], которые во всех случаях одинаковы и поэтому здесь не рассматриваются.

4.1. Линейная и квадратичная интерполяция. В работе [5] для быстрого вычисления элементарных функций в мини-ЭВМ предлагается использовать при $x \in [0,1]$ линейную или квадратичную интерполяцию с равномерным шагом. При линейной интерполяции прямые проводятся через граничные точки подынтервалов, при квадратичной - параболы проводятся через граничные точки и центр подынтервалов. Получающиеся при этом максимальные ошибки приближения приведены в табл.2. Максимальные ошибки приближения равномерными линейными и

Таблица 2

p	Линейная интерполяция				Квадратичная интерполяция			
	$\sin\frac{\pi}{2}x$	$\log_2(1+x)$	$\operatorname{tg}\frac{\pi}{4}x$	2^x	$\sin\frac{\pi}{2}x$	$\log_2(1+x)$	$\operatorname{tg}\frac{\pi}{4}x$	2^x
2	$7\cdot10^{-2}$	$2{,}9\cdot10^{-2}$	$3{,}9\cdot10^{-2}$	$1{,}3\cdot10^{-2}$	$2{,}3\cdot10^{-2}$	$8{,}7\cdot10^{-3}$	$6{,}5\cdot10^{-2}$	$2{,}5\cdot10^{-3}$
4	$1{,}9\cdot10^{-2}$	$9\cdot10^{-3}$	$1{,}3\cdot10^{-2}$	$3{,}4\cdot10^{-3}$	$3{,}6\cdot10^{-3}$	$1{,}7\cdot10^{-3}$	$7{,}3\cdot10^{-3}$	$3{,}2\cdot10^{-4}$
8	$4{,}8\cdot10^{-3}$	$2{,}5\cdot10^{-3}$	$4\cdot10^{-3}$	$9\cdot10^{-4}$	$4{,}8\cdot10^{-4}$	$2{,}7\cdot10^{-4}$	$9{,}2\cdot10^{-4}$	$4\cdot10^{-5}$
16	$1{,}2\cdot10^{-3}$	$6{,}6\cdot10^{-4}$	$1{,}1\cdot10^{-3}$	$2{,}3\cdot10^{-4}$	$6\cdot10^{-5}$	$3{,}9\cdot10^{-5}$	$1{,}2\cdot10^{-4}$	$5{,}1\cdot10^{-6}$
32	$3\cdot10^{-4}$	$1{,}7\cdot10^{-4}$	$2{,}9\cdot10^{-4}$	$5{,}8\cdot10^{-5}$	$7{,}5\cdot10^{-6}$	$6{,}2\cdot10^{-6}$	$1{,}5\cdot10^{-5}$	$6{,}3\cdot10^{-7}$
64	$7{,}5\cdot10^{-5}$	$4{,}3\cdot10^{-5}$	$7{,}3\cdot10^{-5}$	$1{,}5\cdot10^{-5}$	$9{,}3\cdot10^{-7}$	$6{,}8\cdot10^{-7}$	$1{,}9\cdot10^{-6}$	$7{,}9\cdot10^{-8}$
128	$1{,}9\cdot10^{-5}$	$1{,}1\cdot10^{-5}$	$1{,}9\cdot10^{-5}$	$3{,}6\cdot10^{-6}$	$1{,}2\cdot10^{-7}$	$8{,}6\cdot10^{-8}$	$2{,}3\cdot10^{-7}$	$1\cdot10^{-8}$

Таблица 3

p	Линейная интерполяция				Квадратичная интерполяция			
	$\sin\frac{\pi}{2}x$	$\log_2(1+x)$	$\operatorname{tg}\frac{\pi}{4}x$	2^x	$\sin\frac{\pi}{2}x$	$\log_2(1+x)$	$\operatorname{tg}\frac{\pi}{4}x$	2^x
2	$2{,}4\cdot10^{-2}$	$1{,}1\cdot10^{-2}$	$1{,}1\cdot10^{-2}$	$1\cdot10^{-2}$	$2{,}4\cdot10^{-3}$	$9{,}5\cdot10^{-4}$	$1{,}9\cdot10^{-3}$	$4{,}5\cdot10^{-4}$
4	$5{,}8\cdot10^{-3}$	$2{,}7\cdot10^{-3}$	$2{,}6\cdot10^{-3}$	$2{,}7\cdot10^{-3}$	$3{,}5\cdot10^{-4}$	$1{,}4\cdot10^{-4}$	$3\cdot10^{-4}$	$7\cdot10^{-5}$
8	$1{,}4\ 10^{-3}$	$6{,}8\ 10^{-4}$	$6{,}4\ 10^{-4}$	$6{,}7\ 10^{-4}$	$4{,}7\ 10^{-5}$	$2\ 10^{-5}$	$4{,}3\ 10^{-5}$	$9{,}8\ 10^{-6}$
16	$9{,}5\cdot10^{-4}$	$1{,}7\cdot10^{-4}$	$1{,}6\cdot10^{-4}$	$1{,}7\cdot10^{-4}$	$5{,}9\cdot10^{-6}$	$2{,}6\cdot10^{-6}$	$5{,}8\cdot10^{-6}$	$1{,}3\cdot10^{-6}$
32	$8{,}8\cdot10^{-5}$	$4{,}2\cdot10^{-5}$	$3{,}9\cdot10^{-5}$	$4{,}9\cdot10^{-5}$	$7{,}2\cdot10^{-7}$	$3{,}4\cdot10^{-7}$	$7{,}6\cdot10^{-7}$	$1{,}7\cdot10^{-7}$
64	$2{,}2\cdot10^{-5}$	$1\cdot10^{-5}$	$1\cdot10^{-5}$	$1{,}1\cdot10^{-5}$	$9{,}4\cdot10^{-8}$	$4{,}3\cdot10^{-8}$	$9{,}7\cdot10^{-8}$	$2{,}1\cdot10^{-8}$
128	$5{,}6\cdot10^{-6}$	$2{,}5\cdot10^{-6}$	$2{,}5\cdot10^{-6}$	$2{,}6\cdot10^{-6}$	$1{,}2\cdot10^{-8}$	$5{,}3\cdot10^{-9}$	$1\cdot10^{-8}$	$2{,}6\cdot10^{-9}$

квадратичными сплайнами с заданным количеством звеньев p для тех же функций приведены в табл. 3.

Сравнение данных табл. 2 и 3 показывает,что при том же количестве звеньев использование линейных сплайнов дает выигрыш по точности от двух до восьми раз, а использование квадратичных сплайнов - от трех до двадцати раз. При увеличении разбиений выигрыш возрастает. Для достижения той же точности необходимый объем памяти при использовании квадратичных сплайнов уменьшается в два-три раза, для линейных сплайнов - уменьшается незначительно. Сравнение проводится при одинаковом времени вычислений конкретных функций.

4.2. Использование сокращенных таблиц. Метод использования сокращенных таблиц,не требующий проведения вычислений, рассмотрен в работе [6]. Необходимый объем памяти V_k (двоичных разрядов) для вычисления функций $\sin\frac{\pi}{2}x$, e^{x-1}, $\log_2(1+x)$ при $x\in[0,1]$ с точностью $\varepsilon=10^{-k}$, $k=\overline{2,6}$, по этому методу приведен в табл.4 (верхнее число в каждой клетке). Величины получены исходя из данных работы [6].Для сравнения необходимых объемов памяти требуется знать минимальное количество звеньев равномерных сплайн-приближений,обеспечивающих ту же точность. Это количество приведено в табл.5. Необходимый объем памяти для случая линейного сплайна, приближающего функцию f(x) с точностью $\varepsilon = 10^{-k}$, может быть определен по формуле

Таблица 4

ε	$\sin\frac{\pi}{2}x$	e^{x-1}	$\log_2(x+1)$
10^{-2}	576 108	352 54	448 81
10^{-3}	3968 350	2176 245	2944 245
10^{-4}	19456 1364	11766 880	15360 968
10^{-5}	110592 6435	71680 3250	106496 3500
10^{-6}	540672 18910	344232 12338	573720 13144

$$V_k' = p(2L_{k+1} + L_{[k/2+1]}), \qquad (2)$$

где p - количество звеньев сплайна, L_i - количество двоичных разрядов, соответствующих i-десятичным разрядам, [a] - целая часть числа a .

Формула (2) написана исходя из того, что для записи каждого параметра линейного сплайна, вычисленного с точностью 10^{-k}, до-

Таблица 5

ε	Линейные сплайны			Параболические сплайны		
	$\sin\frac{\pi}{2}x$	e^{x-1}	$\log_2(x+1)$	$\sin\frac{\pi}{2}x$	e^{x-1}	$\log_2(x+1)$
10^{-2}	4	2	3	2	I	I
10^{-3}	I0	7	7	3	2	2
10^{-4}	3I	20	22	7	4	5
10^{-5}	99	65	70	I4	9	II
10^{-6}	305	I99	2I2	30	20	23

статочно (k+1) десятичных или L_{k+1} двоичных разрядов, а для записи границ звеньев сплайна достаточно [k/2+1] десятичных или L[k/2+1] двоичных разрядов [8].

Величины V'_k приведены в табл. 4 (нижнее число в каждой клетке). Сравнение величин V_k и V'_k показывает, что при использовании равномерных линейных сплайн-приближений объем необходимой памяти уменьшается от пяти до сорока раз, причем выигрыш в объеме памяти возрастает при увеличении точности вычислений. При этом, однако, тратится некоторое время на выполнение одной операции сложения и одной операции умножения.

Аналогично можно показать, что при использовании параболических равномерных сплайнов объем необходимой памяти при точности вычислений от 10^{-2} до 10^{-6} уменьшается от восьми до трехсот раз.

4.3. Использование отрезков ряда Тейлора и интерполяционных многочленов [4]. Промежуток приближения $[\alpha, \beta]$ разбивается на $p = 2^c$ подынтервалов, на каждом из которых функция приближается многочленом степени m. В работе [4] отмечено, что удобнее использовать отрезок ряда Тейлора. При этом обозначим: M_{m+1} - максимум абсолютной величины (m+1)-й производной приближаемой функции; $g = \log_2(m+1)$, где $\varepsilon = 2^{-n}$ - точность приближения; $l = (\log_2 M_{m+1})/M_{m+1}$; $r = 2^{-(n+p)} \approx 10^{-k+1}$ - предельно допустимая ошибка метода вычисления. Перечисленные величины должны удовлетворять неравенству

$$(c-g)(m+1) + m + 1((m+1)!) + 1 - lM_{m+1} \geq n + p , \qquad (3)$$

из которого при заданных приближаемой функции и точности прибли - жения могут быть определены подходящие величины c и m [4].

Таблица 6

ε	m = 1			m = 2		
	$\sin\frac{\pi}{2}x$	e^{x-1}	$\log_2(x+1)$	$\sin\frac{\pi}{2}x$	e^{x-1}	$\log_2(x+1)$
10^{-2}	128 21	128 43	128 28	32 8	64 32	32 16
10^{-3}	512 34	512 49	512 49	128 21	128 32	128 37
10^{-4}	2048 44	2048 68	2040 62	256 18	512 64	256 26
10^{-5}	8196 55	8196 83	8196 77	2048 73	1024 57	1024 46
10^{-6}	32772 72	32772 11	32772 103.	8196 136	2048 51	2948 46

Проведем сравнение необходимого объема памяти при одинако - вой скорости вычислений. В табл.6 приведено количество p подынтервалов (первая строка в каждой клетке), на которое необходимо разбить промежуток [0,1] для того, чтобы можно было приблизить функции $\sin\frac{\pi}{2}x$, e^{x-1} и $\log_2(1+x)$ с точностью 10^{-2} - 10^{-6} при $m=1$ и $m=2$. Сравнение с данными табл.5 показывает, что использование равномерных линейных и параболических сплайнов вместо отрезков ряда Тейлора позволяет уменьшить объем памяти в десятки раз при той же скорости и точности вычислений. Во второй строке табл.6 приведено отношение соответствующих объемов памяти. При возрастании точности это отношение увеличивается.

Исходя из неравенства (3), можно также попытаться найти количество $m+1$ членов ряда Тейлора, которые необходимы для вычисления перечисленных функций с использованием того же объема памяти, что и в случае приближения равномерными линейными и параболическими сплайнами. Наименьший проигрыш во времени вычисления для случая линейного равномерного сплайна - в три раза - получается

для функции $\sin\frac{\pi}{2}x$ при $\varepsilon = 10^{-5}$ и 10^{-6}, а при $\varepsilon = 10^{-2}$, 10^{-3}, 10^{-4} - проигрыш в четыре раза. Для случая равномерного параболического сплайна и функции $\sin\frac{\pi}{2}x$ - проигрыш в два раза. Для функций e^{x-1} и $\log_2(1+x)$ в ряде случаев объем памяти, соответствующий приближению равномерными линейными и параболическими сплайнами, при приближении отрезками ряда Тейлора с точностью $\varepsilon = 10^{-k}$, $k = \overline{1,6}$, недостижим. В других случаях одинаковый объем памяти получается при $m > 10$, при этом скорость вычислений уменьшается в десять и более раз.

Литература

1. СЕРГИЕНКО И.В., ПОПОВ Б.А. Уточнение класса инженерных задач, решаемых на ЭВМ "Промінь". -В кн.: Математическое обеспечение ЭВМ и эффективная организация вычислительного процесса. Киев, 1969, вып. 2, с.3-17.

2. МОНЦИБОВИЧ Б.Р., ПОПОВ Б.А., СЕРГИЕНКО И.В. Исследование класса инженерных задач, решаемых на ЭВМ "Минск-22" и вопросы сравнения эффективности вычислительных машин. -Киев, Б.и., 1971, - 18 с. (Препринт/ИК АН УССР: 71-38).

3. Computer approximations /J.F.Hart, E.W.Cheney, C.L.Lawson, et al. - New York: Wiley, 1968. - 344 p.

4. Табличные процессоры для ЕС ЭВМ /Е.П.Балашов, С.В. Гусев, В.Н.Негода и др. -В кн.: Гибридные вычислительные машины и комплексы, 1979, вып. 1, с. 30-35.

5. ПАЛАГИН А.В., КУРГАЕВ А.Ф., КОНДРАЧУК И.М. К выбору метода вычисления элементарных функций в мини-ЭВМ. -УСиМ, 1973, №5, с. 65-69.

6. ИЛЬИН В.А., ПОПОВ Ю.А., ДРУЖИНИН И.И. Об использовании сокращенных таблиц при вычислении элементарных функций. -УСиМ, 1979, № 1, с. 58- 60, 86.

7. ПОТАПОВ Б.И., ФЛОРЕНСОВ А.Н. Таблично-алгоритмический метод реализации в ЦВМ функции логарифма. -УСиМ, 1978, №4, с.90-95.

8. ПОПОВ Б.А., ТЕСЛЕР Г.С. Приближение функций для технических приложений. - Киев: Наукова думка, 1980. - 352 с.

9. КУРГАЕВ А.Ф., ПАЛАГИН А.В.Устройство для формирования адресов табличных функций. А.С. № 518770. -Бюл. изобрет.,1976, №23.

10. ФЕТ Я.И. Массовая обработка информации в специализированных однородных процессорах. -Новосибирск: Наука, 1976. - 200 с.

11. AUS H.M., KORN L.A. Table-lookup interpolation function generation for fixedpoint digital computations. - IEEETC, 1969, v. 18, N 8, p. 745-748.

12. БЕРКОВИЧ С.Я., КОЧИН Ю.Я. О поиске чисел, ближайших к заданному. -Авт. и телемех., 1975, № 2, с.176-178.

13. ФЕТ Я.И. Элемент ассоциативной матрицы памяти. А.С. №634372.-Бюл. изобрет., 1978, №43, с.195.

I4. БЛАГОВЕЩЕНСКИЙ Ю.В., ТЕСЛЕР Г.С. Вычисление элементарных функций на ЭВМ. -Киев: Техника, I977. - 208 с.

Поступила в ред.-изд.отд.
23 апреля I98I года