РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
А В Т О М Е Т Р И Я
2003, том 39, № 3

УДК 539.2.536.1

В. П. Маркова
(Новосибирск)

ПРИМЕНЕНИЕ МОДУЛЯРНОЙ АРИФМЕТИКИ ДЛЯ МОДЕЛИРОВАНИЯ ДИФФУЗИИ

Исследуются вычислительные возможности системы счисления в остаточных классах для моделирования процесса одномерной диффузии. Конечно-разностная схема диффузии записана в терминах системы счисления в остаточных классах (СОК). Сравниваются вычислительные характеристики (устойчивость, точность и временная сложность), полученные в результате моделирования процесса диффузии в СОК, с аналогичными характеристиками решения уравнения диффузии конечно-разностным явным методом.

Введение. Быстрое и точное моделирование процесса диффузии есть важная задача, поскольку диффузия является составной частью большого круга физических, химических и биологических явлений, которые традиционно описываются дифференциальными уравнениями в частных производных. Увеличение скорости моделирования процесса диффузии может быть достигнуто за счет выбора таких методов вычислений, которые обладают естественным пространственным параллелизмом и локальностью взаимодействий, т. е. методов, в которых вычисления на каждом шаге по времени и пространству выполняются независимо друг от друга. Именно поэтому мы отдаем предпочтение явным методам. Одним из возможных путей получения точного решения является переход от вещественных чисел к целым. Более того, для выполнения вычислений в целых числах желательно использовать такую арифметику, которая позволит увеличить и точность, и скорость вычислений. В работе для организации вычислений используется широко распространенная в обработке сигналов и изображений система счисления в остаточных классах.

Система счисления в остаточных классах (СОК) [1–5] принадлежит к непозиционным числовым системам и представляет любое целое число в виде множества остатков по модулям из множества попарно взаимно простых чисел. В отличие от позиционных числовых систем все остатки модульного представления независимы и являются малоразрядными числами. Эти свойства модульного представления позволяют свести выполнение основных арифметических операций (сложения, вычитания и умножения) к параллельному модульному и быстрому выполнению одноименных операций для каждого модуля. Более того, вычисления в остаточных классах свободны от ошибок округления. К сожалению, операции деления, определения знака,

сравнения в общем случае выполняются неэффективно. Таким образом, использование СОК позволяет ввести дополнительный уровень параллелизма в реализацию алгоритма: параллелизм на уровне модулей. Конечно, максимальное быстродействие такие алгоритмы могут демонстрировать на процессорах с мультитредовой архитектурой, если количество поддерживаемых тредов близко или равно количеству модулей СОК, либо на алгоритмически ориентированных процессорах, архитектура которых поддерживает параллелизм на уровне модулей и разрядов.

В работе исследуются вычислительные возможности системы счисления в остаточных классах на примере моделирования процесса одномерной диффузии явным методом. В связи с трудностями деления целых чисел в СОК в реализацию традиционного конечно-разностного представления уравнения диффузии введены два приема: 1) перенос остатка от деления на следующий шаг, 2) представление обратного диффузионного числа в виде отношения целых чисел. Предложен алгоритм деления на один из модулей, временная сложность (количество шагов, необходимое для реализации алгоритма) которого вдвое меньше известного алгоритма из [1]. Выполнено численное моделирование процесса диффузии на Pentium III. Результаты экспериментов показали, что вычисления в СОК устойчивы на большем диапазоне величины обратного диффузионного числа. Решение по точности не уступает решению в вещественных числах. Как и следовало ожидать, временная сложность моделирования процесса диффузии в СОК на традиционных компьютерах превосходит временную сложность моделирования в вещественных числах.

Работа состоит из двух разделов. В разд. 1 дается краткое представление о модульной арифметике. В разд. 2 описаны конечно-разностное представление диффузии в СОК, результаты моделирования процесса диффузии на Pentium III и архитектура процессора.

1. **Модульная арифметика.** 1.1. *Модульное представление.* Основой системы счисления в остаточных классах является множество попарно простых чисел (модулей) $\mathrm{p}=\{p_0, p_1, \ldots, p_{k-1}\}$ таких, что $\mathrm{НОД}(p_i, p_j)=1$ для $i \neq j$, где $\mathrm{НОД}(p_i, p_j)$ – наибольший общий делитель модулей p_i и p_j. Известно [1–5], что любое целое число $X \in [0, P)$, $P=\prod_{j=0}^{k-1} p_j$, имеет единственное представление в виде

$$X \to (X_0, X_1, \ldots, X_{k-1}) = \mathbf{X}, \qquad (1)$$

где $X_j = X \bmod p_j$ – j-й остаток (вычет) целого X по модулю p_j. Множество вычетов (1) называется модульным представлением числа X. Интервал $[0, P)$ определяет динамический диапазон представления целых положительных чисел. Остаток X_j вычисляется следующим образом:

$$X \bmod p_j = X_j = X - \lfloor X / p_j \rfloor p_j,$$

где $\lfloor Y \rfloor$ обозначает наибольшее целое, меньшее или равное Y. Из определения модульного представления числа видны следующие его свойства:

– все остатки независимы, т. е. переносы между остатками отсутствуют,

– длина каждого остатка меньше длины исходного числа.

Эти свойства модульного представления обеспечивают параллельную модульную и быструю арифметику.

При построении системы счисления в остаточных классах одной из важных задач является выбор системы модулей, так как она существенно влияет на длину динамического диапазона представления чисел, скорость выполнения операций, сложность преобразования из обычного представления в модульное и наоборот.

1.2. *Модульная арифметика.* Пусть $\mathrm{p}=\{p_0, p_1, \ldots, p_{k-1}\}$ – множество попарно взаимно простых модулей. $\mathbf{X}=(X_0, X_1, \ldots, X_{k-1})$ и $\mathbf{Y}=(Y_0, Y_1, \ldots, Y_{k-1})$ – модульные представления целых чисел X и Y, $X \in [0, P), Y \in [0, P)$. Тогда модульное представление целого числа $Z = X \oplus Y$, $Z \in [0, P)$, определяется следующим образом:

$$\mathbf{Z}=(Z_0, Z_1, \ldots, Z_{k-1})=\mathbf{X} \oplus \mathbf{Y}, \qquad (2)$$

где $Z_j=(X_j \oplus Y_j) \bmod p_j$ для всех $j=0,1,\ldots,k-1$; символ $\oplus$ обозначает сложение, вычитание или умножение.

Пример 1. Пусть $\mathrm{p}=\{3,5,7,11\}$, $\mathbf{X}=(2,4,3,4)$, $\mathbf{Y}=(2,4,0,3)$ ($X_{10}=59$, $Y_{10}=14$). Тогда согласно (1) $\mathbf{Z}=(1,3,3,7)$, если $Z=X+Y$, $\mathbf{Z}=(0,0,3,1)$, если $Z=X-Y$, и $\mathbf{Z}=(1,1,0,1)$, если $Z=X \times Y$.

Выражение (2) демонстрирует параллельную и модульную природу выполнения основных операций в СОК. Каждая из этих операций выполняется за одну модульную операцию. Скорость выполнения модульной операции в n/m раз выше скорости выполнения одноименной двоичной операции, где n – разрядность чисел в двоичном представлении, а m – разрядность максимального остатка. Более того, вычисления в системе остаточных классов свободны от ошибок округления. Однако в системе есть так называемые немодульные операции (деление, определение знака, сравнение), которые выполняются достаточно сложно. Приведем частный случай деления на один из модулей СОК.

Деление. Пусть $\mathrm{p}=\{p_0, p_1, \ldots, p_{k-1}\}$ – множество попарно взаимно простых модулей, расположенных в возрастающем порядке. $\mathbf{X}=(X_0, X_1, \ldots, X_{k-1})$ – делимое, $\mathbf{p}_i=(\pi_0, \ldots, \pi_{i-1}, 0, \underbrace{p_i, \ldots, p_i}_{k-1-i})$ – делитель, где $\pi_j = p_i \bmod p_j$ для $j=0,\ldots,i-1$. Если число X делится нацело на модуль p_0, то $X_0=0$, в противном случае в качестве делимого берется число $X'=X-X_i$. Тогда деление выполняется в два шага.

На первом шаге алгоритм формирует первое приближение частного $\mathbf{U}=(Z_0, \ldots, Z_{i-1}, 0, Z_{i+1}, \ldots, Z_{k-1})$. Чтобы избежать неопределенности, остаток Z_i приравнивается к нулю. Остаток Z_j, $j \neq i$, вычисляется поразрядным делением остатка X_j на соответствующий остаток делителя (π_j или p_i). Если остаток X_j не делится нацело на остаток π_j или модуль p_i, тогда деление заменяется умножением:

$$\frac{X_j}{\pi_j} \bmod p_j \rightarrow (X_j a_j) \bmod p_j,$$

где a_j – обратная мультипликативная величина делителя, т. е. $a_j \pi_j \bmod p_j \equiv 1$.

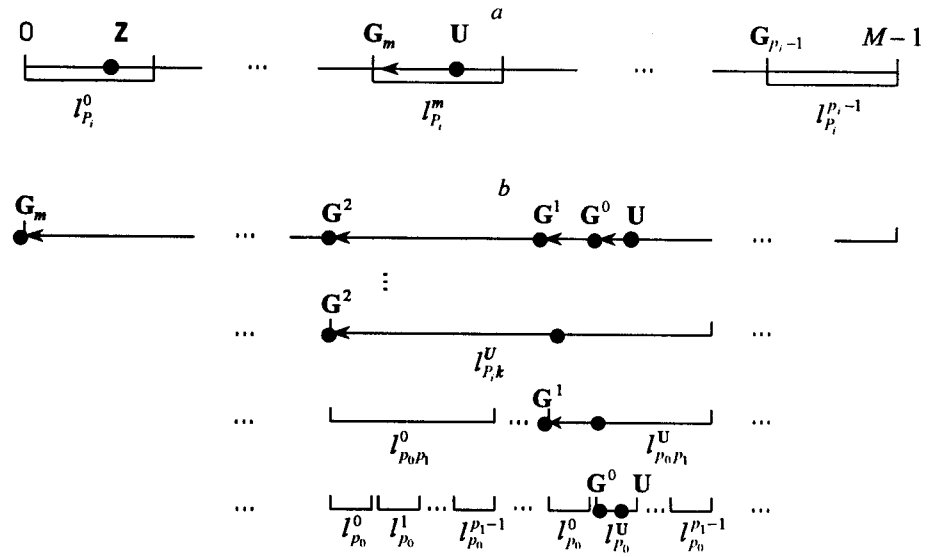

Рис. 1. Геометрическое представление внутренних интервалов (*a*) и нулевизации (*b*)

На втором шаге алгоритм определяет значение остатка Z_i. Предлагается очень простой алгоритм вычисления его значения. Итак, первое приближение частного **U** может принадлежать одному из интервалов $l^m_{P_i}$, $P_i = P/p_i$, $m = 0, \ldots, p_i - 1$, в диапазоне $[0, P)$ (рис. 1, *a*). Далее эти интервалы будем называть внутренними. (Интервал $l^m_{P_i}$, которому принадлежит первое приближение частного, назовем **U**-интервалом и обозначим $l^{\mathbf{U}}_{P_i}$.) Внутренние интервалы образуются в результате разбиения динамического диапазона $[0, P)$ на p_i частей. Каждый интервал $l^m_{P_i}$ включает P_i чисел и представляет собой множество вложенных интервалов разной длины (рис. 1, *b*). Наименьший интервал l_{p_0} состоит из p_0 чисел, p_1 таких интервалов образуют интервал $l_{p_0 p_1}$. Следующий по величине интервал $l_{p_0 p_1 p_2}$ состоит из p_2 интервалов $l_{p_0 p_1}$ и т. д. Первое число m-го внутреннего интервала имеет следующий вид:

$$\mathbf{G}_m = (0, 0, \ldots, 0, \underbrace{m}_{i}, 0, \ldots, 0),$$

где $G_{mj} = 0$ для всех $j \neq i$, так как число G_m кратно каждому модулю p_j. Частное **Z** в отличие от приближения частного **U** всегда принадлежит интервалу $[0, P)$, так как $Z \leq P_i$. Согласно организации внутреннего интервала оба числа **Z** и **U** расположены на одном и том же расстоянии от начал соответствующих им внутренних интервалов, т. е.

$$\mathbf{Z} - (0, 0, \ldots, 0) = \mathbf{U} - \mathbf{G}_m. \qquad (3)$$

Для i-го остатка соотношение (3) может быть переписано как $Z_i - 0 = (0 - m) \bmod p_i$, т. е. $Z_i = (0 - m) \bmod p_i$. Следовательно, вычисление остатка Z_i сводится к нахождению первого числа **U**-интервала, которое определяется в

результате выполнения преобразования $\mathbf{U} \to \mathbf{G}_m$, называемого нулевизацией [1].

Нулевизация состоит в последовательном увеличении количества нулевых остатков в модульном представлении преобразуемого числа (исходного **U** или промежуточного), при этом преобразуемое число не должно выходить из соответствующего ему интервала. Нулевизация выполняется за $(k-1)$ шагов и начинается с наименьшего ненулевого остатка, исключая i-й остаток. Процесс преобразования может быть представлен следующим образом:

$$(Z_0, \ldots, Z_{i-1}, 0, Z_{i+1}, \ldots, Z_{k-1}) = \mathbf{U} \to (0, G_1^0, \ldots, G_{k-1}^0) =$$

$$= \mathbf{G}^0 \to (0, 0, G_2^1, \ldots, G_{k-1}^1) = \mathbf{G}^1 \to \ldots \to \mathbf{G}_m.$$

На первом шаге нулевизации по числу **U** определяется число $\mathbf{G}^0$, у которого нулевой остаток равен нулю. В общем случае нулевые значения могут принимать некоторые ненулевые остатки числа $\mathbf{G}^0$. На втором шаге нулевизации по числу $\mathbf{G}^0$ находится число $\mathbf{G}^1$, у которого уже два остатка (нулевой и первый) равны нулю. Процесс продолжается до тех пор, пока не будет определено число $\mathbf{G}_m$. Числа $\mathbf{G}^0, \mathbf{G}^1, \ldots, \mathbf{G}_m$, получаемые последовательно на каждом шаге нулевизации, являются первыми числами интервалов $l_{P_0}^{\mathbf{U}}$ (интервал l_{P_0}, которому принадлежит число **U**), $l_{P_0P_1}^{\mathbf{U}}, \ldots, l_{P_i}^{\mathbf{U}}$, $l_{P_0}^{\mathbf{U}} \subset l_{P_0P_1}^{\mathbf{U}} \subset \ldots \subset l_{P_i}^{\mathbf{U}}$. Геометрически преобразование $\mathbf{U} \to \mathbf{G}_m$ можно рассматривать как последовательное движение из числа **U** в число $\mathbf{G}_m$ через начала вложенных интервалов, начиная с наименьшего интервала $l_{P_0}^{\mathbf{U}}$, до тех пор, пока не достигнем начала **U**-интервала.

Объясним технику нулевизации. Она представляет собой итерационную процедуру. Пусть число $\mathbf{G}^t = (0, \ldots, 0, G_j^t, \ldots, G_{k-1}^t)$ – результат t-го шага нулевизации, $t = 0, 1, \ldots, k-2$. Если $t < i$, то $j = t$, иначе $j = t+1$. Тогда число $\mathbf{G}^{t+1} = \mathbf{G}^t - \mathbf{M}_{n_j}^j$, где число $\mathbf{M}_{n_j}^j = (0, \ldots, 0, n_j, \ldots, M_{k-1}^j)$, $n_j = G_j^t$. Число $\mathbf{M}_{n_j}^j$ называется константой нулевизации. Каждый модуль p_j имеет $(p_j - 1)$ константу нулевизации. Первое число процесса нулевизации – $\mathbf{G}^0 = \mathbf{U} - \mathbf{M}_{P_0}^0$, где $\mathbf{M}_{P_0}^0 = (n_0, M_1^0, \ldots, M_{k-1}^0)$, $n_0 = Z_0$, последнее число – $\mathbf{G}_m = \mathbf{G}^{k-2}$. Константы нулевизации определяются заранее.

Предложенный алгоритм вычисления остатка Z_i имеет меньшую временную сложность по сравнению с известным алгоритмом из [1]. Большая сложность алгоритма Акушского и Юдицкого связана с тем, что этот алгоритм, в отличие от предложенного, работает на интервалах, полученных в результате разбиения динамического диапазона на количество частей, равное максимальному модулю СОК. Этот модуль по условию алгоритма не является делителем. В результате такого разбиения номер интервала, в котором лежит частное, неизвестен. Поэтому в алгоритм введен дополнительный шаг, определяющий номер интервала, которому принадлежит делимое. Для реализации этого шага используется техника нулевизации, следовательно, временная сложность дополнительного шага равна сложности алгоритма определения интервала, которому принадлежит приближение частного.

Временная сложность операции деления $T_д$ определяется следующим образом. Первый шаг алгоритма в худшем случае выполняется за $((k-1)T_{обр}+T_у)$ тактов, где $T_{обр}$ – время обращения к таблице за обратными мультипликативными величинами (или константами нулевизации); $T_у$ – время одного умножения. Второй шаг алгоритма требует $((k-1)T_{обр}+(k-1)T_с)$ тактов, где $T_с$ – время одного вычитания (сложения). В результате

$$T_д = 2(k-1)T_{обр} + (k-1)T_с + T_у.$$

Пример 2. Пусть $\mathrm{p}=\{3,5,7,11\}$, $\mathbf{X}=(2,0,4,8)$ $(X_{10}=305)$ – делимое, $\mathbf{p}_1=(2,0,5,5)$ – делитель, $P=1155$. Тогда остатки, исключая первый, определяются как

$$Z_0 = \frac{2}{2}\bmod 3 = \left(2\cdot\left(\frac{1}{2}\right)\bmod 3\right)\bmod 3 = (2\cdot 2)\bmod 3 = 1,$$

$$Z_2 = \frac{4}{5}\bmod 7 = \left(4\cdot\left(\frac{1}{5}\right)\bmod 7\right)\bmod 7 = (4\cdot 3)\bmod 7 = 5,$$

$$Z_3 = \frac{8}{5}\bmod 11 = \left(8\cdot\left(\frac{1}{5}\right)\bmod 11\right)\bmod 11 = (8\cdot 9)\bmod 11 = 6.$$

Таким образом, $\mathbf{U}=(1,0,5,6)$.

Далее выполняется нулевизация числа **U**. Константы нулевизации для модулей $p_0=3$, $p_2=7$, $p_3=11$ приведены в таблице, процесс нулевизации по-

$p_0=3$	$p_2=7$	$p_3=11$
$\mathbf{M}_0^0=(0,0,0,0)$	$\mathbf{M}_0^2=(0,0,0,0)$	$\mathbf{M}_0^3=(0,0,0,0)$
$\mathbf{M}_1^0=(1,1,1,1)$	$\mathbf{M}_1^2=(0,0,1,4)$	$\mathbf{M}_1^3=(0,0,0,1)$
$\mathbf{M}_2^0=(2,2,2,2)$	$\mathbf{M}_2^2=(0,4,2,9)$	$\mathbf{M}_2^3=(0,4,0,2)$
	$\mathbf{M}_3^2=(0,3,3,3)$	$\mathbf{M}_3^3=(0,3,0,3)$
	$\mathbf{M}_4^2=(0,3,4,7)$	$\mathbf{M}_4^3=(0,2,0,4)$
	$\mathbf{M}_5^2=(0,2,5,1)$	$\mathbf{M}_5^3=(0,1,0,5)$
	$\mathbf{M}_6^2=(0,1,6,6)$	$\mathbf{M}_6^3=(0,0,0,6)$
		$\mathbf{M}_7^3=(0,4,0,7)$
		$\mathbf{M}_8^3=(0,3,0,8)$
		$\mathbf{M}_9^3=(0,2,0,9)$
		$\mathbf{M}_{10}^3=(0,1,0,10)$

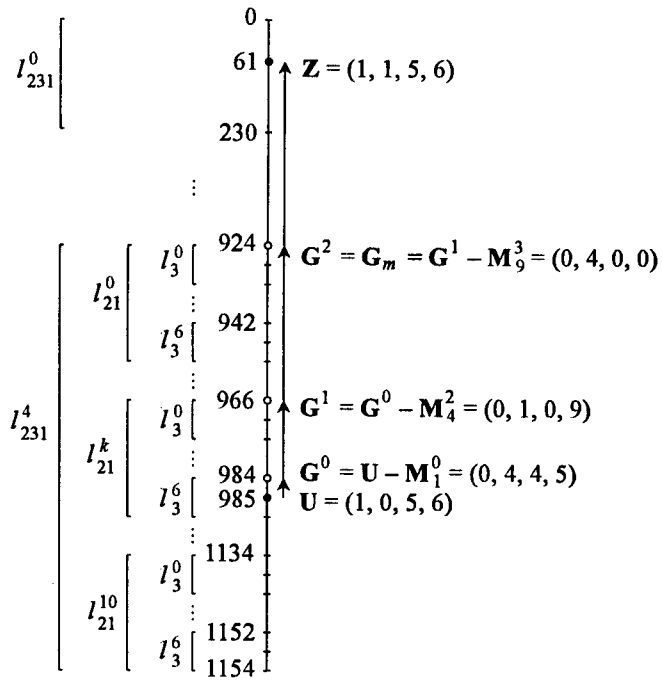

Рис. 2. Пример деления в СОК

казан на рис. 2. Для $P = 3 \times 5 \times 7 \times 11$ и модуля p_1 в качестве делителя диапазон $[0, P)$ разбивается на 5 внутренних интервалов l_{231}. Каждый интервал l_{231}, состоящий из 231 числа, разбивается на 11 интервалов l_{21}. Минимальный интервал l_3 включает 3 числа, 7 таких интервалов образуют интервал l_{21}.

Для того, чтобы нулевой остаток числа $\mathbf{G}^0$ стал равен нулю, из таблицы выбирается константа, соответствующая модулю p_0, $\mathbf{M}_1^0 = (1,1,1,1)$, так как $Z_0 = 1$ и вычитается из числа **U**. В результате $\mathbf{G}^0 = (0, 4, 4, 5)$. Второй остаток числа $\mathbf{G}^1$ устанавливает в нуль константа $\mathbf{M}_4^2 = (0,3,4,7)$, и тогда $\mathbf{G}^1 = (0,1,0,9)$. И наконец, число $\mathbf{G}^2$, равное $\mathbf{G}^1 - \mathbf{M}_9^3 = (0,4,0,0)$, указывает, что число **U** принадлежит 4-му внутреннему интервалу, т. е. $4 \cdot 231 \le (1,0,5,6) \le 5 \cdot 231$. Следовательно, $Z_2 = (0-4) \bmod 5 = 1$ и частное $\mathbf{Z} = (1,1,5,6)$ $(Z_{10} = 61)$.

2. **Моделирование уравнения диффузии в СОК.** 2.1. *Уравнение диффузии.* Известно, что уравнение одномерной диффузии

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \qquad u(x)\big|_{t=0} = f(x),$$

где u – плотность (концентрация); α – коэффициент диффузии; x – координата одномерного пространства; $f(x)$ – начальное условие в результате использования явной схемы дискретизации пространства и времени при $x = ih$, $t = m\tau$ (где $i = 0,1,\ldots, N-1$, N – количество узлов пространства; $m = 0,1,\ldots$; h и τ – шаги по пространству и времени), имеет следующий вид:

$$u_i^{m+1} = u_i^m + \frac{1}{D}(u_{i-1}^m + u_{i+1}^m - 2u_i^m) = u_i^m + \frac{1}{D}L(u_i^m). \qquad (4)$$

Здесь u_i^m – значение функции в узле одномерной сетки; $L(u_i^m) = u_{i-1}^m + u_{i+1}^m - 2u_i^m$; $\frac{1}{D} = \frac{\alpha\tau}{h^2}$. Величина $\frac{1}{D}$ называется диффузионным числом. Соотношение (4) задает итерационный алгоритм решения уравнения диффузии. Далее решение, полученное в соответствии с формулой (4), в которой значения функции в узлах сетки и начальные условия являются вещественными числами, будем называть решением в вещественных числах.

2.2. *Реализация уравнения диффузии в СОК.* Рассмотрим уравнение (4), в котором значения функции в узлах сетки и начальные условия являются целыми числами из интервала $[0, P)$ и вычисления выполняются в СОК с множеством модулей $\mathrm{p} = \{p_0, p_1, \ldots, p_{k-1}\}$, $P = \prod_{i=0}^{k-1} p_i$. Чтобы исключить потерю точности, связанную с делением целых чисел в СОК, введем следующие два приема в конечно-разностное представление уравнения диффузии.

Представление обратного диффузионного числа в виде отношения целых чисел. Поскольку диффузионное число является дробью по определению, представим его в виде несократимой дроби $\frac{1}{D} = \frac{D_1}{D_2}$, где D_1 и D_2 – целые числа. Тогда уравнение (4) имеет следующий вид:

$$u_i^{m+1} = u_i^m + \frac{D_1 L(u_i^m)}{D_2}. \tag{5}$$

Очевидно, что временная сложность вычисления u_i^{m+1} будет минимальной, если делитель D_2 является модулем множества p. Из этого следует, что один из модулей множества должен быть равен числу D_2, $D_2 = p$. Если $D_2 = 2^n$ или $D_2 = 2^n \pm 1$ и диапазон представления чисел равен произведению $(2^n - 1) \times 2^n \times (2^n + 1)$, то в качестве множества модулей следует выбирать множества вида $\{2^n - 1, 2^n, 2^n + 1\}$, которые очень популярны в цифровой обработке сигналов [4, 5]. Если D_2 – очень большое число и $D_2 = D_3 \times D_4$, то модулями множества p должны быть числа D_3 и D_4.

Перенос остатка на следующий шаг. Представим произведение $D_1 L(u_i^0)$ в виде суммы $L_i^0 + R_i^1$, где $L_i^0 = \left\lfloor \frac{D_1 L(u_i^0)}{p} \right\rfloor p$, $R_i^1 = (D_1 L(u_i^0)) \bmod p$. Тогда $u_i^1 = u_i^0 + \frac{L_i^0}{p}$. Остаток R_i^1 переносится на следующий шаг и прибавляется к делимому $D_1 L(u_i^1)$ и т. д. Согласно данному приему соотношение (5) можно представить в виде

$$u_i^{m+1} = u_i^m + \frac{L_i^m}{p}, \qquad L_i^m = \left\lfloor \frac{D_1 L(u_i^m) + R_i^m}{p} \right\rfloor p,$$
$$R_i^m = (D_1 L(u_i^{m-1}) + R_i^{m-1}) \bmod p, \qquad R_i^0 = 0. \tag{6}$$

Поскольку в работе СОК определена только для положительных чисел, то на каждой итерации необходимо определять знак величин $Q_i^m = D_1 L(u_i^m) + R_i^m$ и $R_i^m = Q_i^{m-1} \bmod p$. В зависимости от знака величины Q_i^m уравнение (6) имеет следующий вид:

$$u_i^{m+1} = u_i^m + \operatorname{sign}(Q_i^m) \left\lfloor \frac{V_i^m}{p} \right\rfloor p, \tag{7}$$

где $V_i^m = Q_i^m$, если $Q_i^m \geq 0$, и $V_i^m = P - Q_i^m$ в противном случае. Для определения знака используется алгоритм [6]. Решение, полученное в соответствии с формулой (7), будем называть решением в СОК.

2.3. *Результаты моделирования.* Для исследования характеристик вычислений в СОК (устойчивости, точности и временной сложности) выполнено численное моделирование процесса диффузии на Pentium III. Эксперименты проводились для СОК с множеством оснований $p = \{5, 7, 11, 13\}$ и гладкой функции в качестве начального распределения. Функция определена на интервале $[0, 90)$ и имеет следующий вид:

$$f(x) = \begin{cases} 1000\cos(x) + 1000, & \text{если } 0 \leq x < 40, \\ 0, & \text{если } 40 \leq x < 90. \end{cases}$$

Для сравнения в качестве стандартного решения использовалось вещественное решение уравнения диффузии конечно-разностным явным методом.

1. Устойчивость вычислений в СОК. Известно [7], что вычисление уравнения одномерной диффузии в вещественных числах явным конечно-разностным методом устойчиво, если $D \geq 2$. Здесь исследование устойчивости сводится к экспериментальному определению величины обратного диффузионного числа D, при котором вычисление уравнения диффузии в СОК устойчиво для заданного исходного распределения. Для этого строится функция зависимости дисперсии амплитуды плотности от времени для различных значений D.

Эксперименты показали, что вычисления в СОК устойчивы для $D > 1{,}64$. На рис. 3 приведен график зависимости дисперсии амплитуды плотности от времени для решений уравнения диффузии в СОК и в вещественных числах

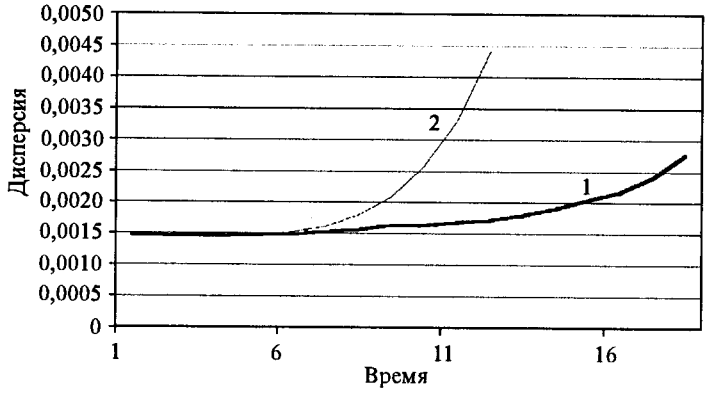

Рис. 3. Зависимость дисперсии амплитуды плотности от времени для вычислений в СОК (кривая 1) и в вещественных числах (кривая 2) для $u(t = 0) = f(x)$ и $D = 1{,}64$

Рис. 4. Решения в СОК (кривая 1) и в вещественных числах (кривая 2) уравнения диффузии и начального распределения (кривая 3) на 28-м шаге моделирования для $u(t = 0) = f(x)$ и $D = 1{,}65$

для $D = 1{,}64$. Здесь оба вычисления неустойчивы. В отличие от вычислений в СОК неустойчивость вычислений в вещественных числах нарастает быстрее. Два решения уравнения диффузии для $D = 1{,}65$ показаны на рис. 4. Для заданного D вычисления в СОК устойчивы, а вычисления в вещественных числах неустойчивы.

Таким образом, для заданного начального распределения вычисления в СОК устойчивы на более широком диапазоне величин обратного диффузионного числа D по сравнению с вычислениями в вещественных числах. Это позволяет увеличить шаг по пространству и (или) выполнить деление не на каждом шаге алгоритма, что несколько уменьшит временную сложность алгоритма.

2. Точность вычислений в СОК. Очевидно, что вычисление уравнения диффузии в СОК обеспечивает приемлемую точность решения, поскольку в схеме исключены ошибки, связанные с реализацией деления, и отсутствуют ошибки округления. Проведенные эксперименты показали, что решения в СОК и в вещественных числах для обоих начальных распределений практически совпадают. Чтобы оценить степень их совпадения, введена функция, равная разности двух решений. Относительная ошибка решения (рис. 5) на

Рис. 5. Разница решений в СОК и в вещественных числах на 1000-м шаге моделирования для $u(t = 0) = f_1(x)$ и $D = 2{,}0$

1000-м шаге моделирования составила не более 0,1 % для гладкого начального распределения и $D = 2{,}0$.

3. Временная сложность реализации алгоритма решения уравнения диффузии в СОК. Как и следовало ожидать, временная сложность модульного решения уравнения диффузии на компьютере общего назначения выше сложности традиционного вычисления. Некоторое уменьшение временной сложности (чуть более 20 %) получено на двухпроцессорном компьютере с использованием OpenMP. Существенное улучшение оценки временной сложности можно достичь на мультитредовых процессорах, если количество поддерживаемых тредов близко к количеству модулей СОК, или на алгоритмически ориентированных процессорах, которые поддерживают параллелизм на уровне модулей и разрядов внутри каждого остатка. Архитектура процессора для моделирования уравнения диффузии в СОК приведена на рис. 6. Она соответствует уравнению (7). Для уменьшения временной сложности определение знака и вычисление величин u_i^{m+1} для положительного и отрицательного значений величины Q_i^m выполняются параллельно. Оценка временной сложности вычисления одной итерации

$$T = 5T_{\text{с}} + T_{\text{у}} + T_{\text{д}} = 2(k-1)T_{\text{обр}} + (k+4)T_{\text{с}} + 3T_{\text{у}}.$$

Для моделирования уравнения диффузии в одномерном пространстве потребуется N таких процессоров, если нет ограничений на их количество.

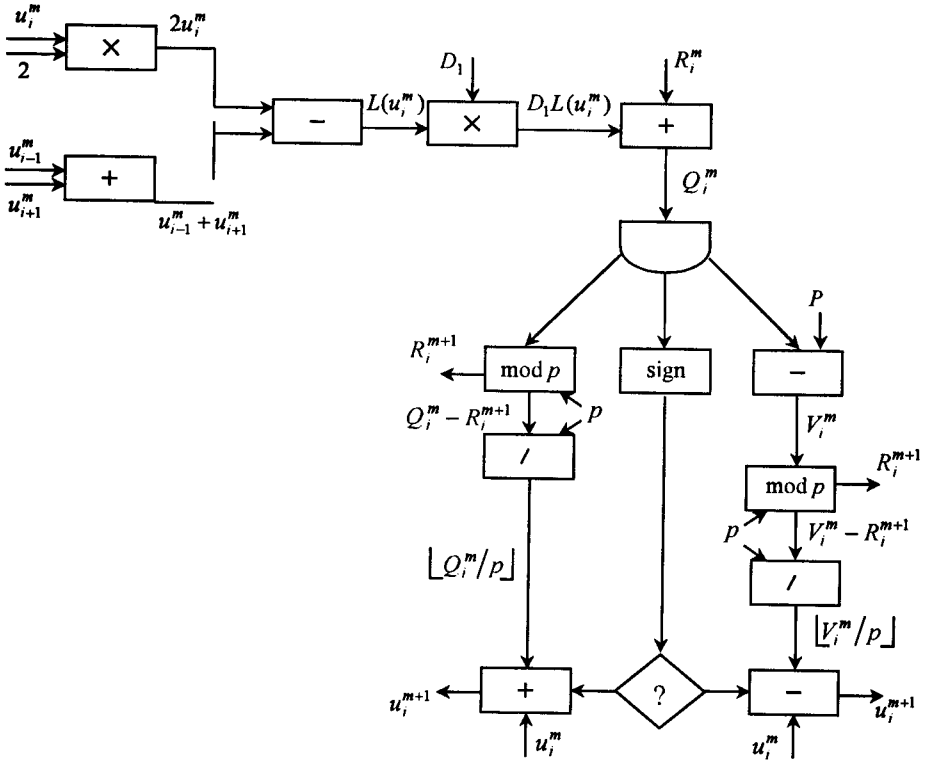

Рис. 6. Архитектура процессора, реализующего уравнение диффузии в СОК

Заключение. В работе предложен алгоритм решения уравнения диффузии в СОК. Чтобы исключить потерю точности при делении целых чисел, в реализацию конечно-разностного представления уравнения диффузии введены два приема: перенос остатка от деления на следующий шаг и представление диффузионного числа в виде отношения двух целых чисел. Проведено численное моделирование процесса диффузии. Результаты моделирования позволяют сделать следующие выводы.

1. Вычисления в СОК устойчивы на более широком диапазоне значений обратного диффузионного числа по сравнению с вычислениями в вещественных числах.

2. Вычисления в СОК обеспечивают приемлемую точность решения.

3. Число D_2 (делитель) само либо его сомножители должно быть включено в множество модулей СОК.

4. Минимальная временная сложность алгоритма вычисления диффузии может быть достигнута только на алгоритмически ориентированных процессорах, которые поддерживают параллелизм на уровне модулей и разрядов.

СПИСОК ЛИТЕРАТУРЫ

1. **Акушский И. Я., Юдицкий Д. И.** Машинная арифметика в остаточных классах. М.: Сов. радио, 1968.
2. **Торгашев В. А.** Система остаточных классов и надежность ЦВМ. М.: Сов. радио, 1973.
3. **Taylor F. J.** Residue arithmetic: a tutorial with examples // IEEE Comput. Mag. 1984. **17**, N 5. P. 50.
4. **Taylor F. J., Huang C. H.** An autoscale residue multiplier // IEEE Trans. Comput. 1982. **C-31**, N 4. P. 321.
5. **Skavantzos A., Adbdallah M.** Implementation issue of the two-level residue number system with pairs of conjugate moduli // IEEE Trans. Signal Processing. 1999. **47**, N 3. P. 826.
6. **Vu T. V.** Efficient implementation of the Chinese remainder theorem for sign detection and residue decoding // IEEE Trans. Comput. 1985. **34**, N 7. P. 646.
7. **Роуч П.** Вычислительная гидродинамика. М.: Мир, 1980.

Институт вычислительной математики и математической геофизики СО РАН,
E-mail: markova@ssd.sscc.ru

Поступила в редакцию
25 марта 2003 г.