КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ

УДК 004.942

ИССЛЕДОВАНИЕ НЕМОДУЛЬНЫХ ОПЕРАЦИЙ В СИСТЕМЕ ОСТАТОЧНЫХ КЛАССОВ

А.Н. ЛАВРИНЕНКО
Н.И. ЧЕРВЯКОВ

Ставропольский государственный университет

e-mail: GreatAntonchik-r@yandex.ru, k-fmf-primath@stavsu.ru

В статье предложено математическое описание ряда методов вычисления позиционных характеристик модулярного кода, а также рассмотрены различные аспекты их применения при выполнении немодульных операций над числами в системе остаточных классов. Разработана программа, моделирующая выполнение этих алгоритмов на ЭВМ и позволяющая производить сравнение временных затрат.

Ключевые слова: система остаточных классов, немодульные операции, компьютерное моделирование.

Система остаточных классов (СОК) по праву служит основой проектирования параллельных вычислительных систем. Операции над числами в СОК делятся на две группы – *модульные*, когда действия над числами можно проводить независимо в параллельных каналах, и *немодульные*, когда для выполнения операции необходима информация обо всем числе.

К немодульным операциям в СОК относится: деление, расширение, определение знака, сравнение, определение переполнения, масштабирование, определение ошибки, локализация ошибки, вычисление ранга и др. Многие из этих операций могут быть реализованы на основе восстановления числа по его остаткам, выполнения необходимых действий и обратного перевода в СОК. Однако такой подход противоречит идее распараллеливания вычислительного процесса.

Анализ существующих алгоритмов выполнения немодульных операций в СОК показывает, что все операции данного класса можно разделить на простые и составные. К *простым* немодульным операциям будем относить те из них, которые могут использоваться как самостоятельные автономные звенья более сложных немодульных операций, называемых, в свою очередь, *составными*.

Деление в СОК рассматривается с двух позиций — как деление нацело и деление с остатком.

В первом случае деление без остатка можно осуществить путем модульного умножения делимого на мультипликативную обратную величину делителя $\left(\left|\frac{b}{a}\right|_{p_i} = \left|b \cdot \frac{1}{a}\right|_{p_i}\right)$.

Исключение составляет только случай когда делитель или его сомножители являются основаниями СОК. При этом цифры частного по всем каналам, кроме совпавших, определяются по общему правилу, как указано выше, а остатки по соответствующим совпавшим

модулям должны быть доопределены путем расширения остаточного представления в СОК.

Во втором случае предлагается алгоритм деления с округлением. Для реализации данного алгоритма необходимо определить остаток α_{n+1} делимого по делителю и модульно вычесть этот остаток, в результате чего будет получена ситуация, аналогичная изложенной в первой части алгоритма [1, 2].

В качестве альтернативы указанных методов можно предложить способ деления путем замены делителя его обратной величиной, эффективно реализуемый на примере действий над обыкновенными дробями с раздельным представлением в СОК числителей и знаменателей. Данный метод обеспечивает высокую точность вычислений, но он более сложен в аппаратной реализации и требует логической модификации существующих алгоритмов.

Алгоритм деления с остатком в СОК по своей сути является частью алгоритма выполнения другой не менее важной немодульной операции — **масштабирования**. Разница между этими двумя операциями состоит лишь в том, что масштабирование в СОК должно применяться к более общему случаю, например, к делению чисел представленных в формате с плавающей точкой (ЧПТ), когда операции в СОК производятся непосредственно над мантиссами. Для ЧПТ характерно использование порядка числа, за счет чего можно корректировать относительную величину числа (масштабировать число, используя масштабный коэффициент) и разрешить проблему деления чисел в СОК в том случае, когда делимое меньше делителя, а также добиться минимальной ошибки округления.

Выполняя деление или масштабирование чисел в СОК, неизбежно придется проводить их **сравнение**. Операция сравнения чисел в СОК более сложна, чем в позиционных системах счисления.

Существует *интервальный метод сравнения* [1, 2], когда в качестве показателя величины числа $A = p_i l_A + \alpha_i$ используется номер интервала

$$l_A = \left| \sum_{i=1}^{n} \left| l_{A_i} \alpha_i \right|_{P_i} \right|_{P_i}, \quad (1)$$

где $l_{A_j} = \frac{P_j^{\varphi(p_j)}}{p_i}, j \neq i$ и $l_{A_i} = \frac{P_i^{\varphi(p_i)} - 1}{p_i}$ — постоянные коэффициенты, определенные системой оснований, $P_i = \frac{P}{p_i}$, α_i – цифры СОК-представления *A*. При таком подходе сравнение чисел сводится к вычислению и сравнению номеров интервалов. В случае если числа принадлежат одному интервалу, предлагается вычислить их разность и по знаку этой разности сделать вывод о сравнении чисел. Для данного метода большое значение имеет выбор дробящего модуля p_i. Чем меньше p_i, тем на большее число интервалов $\left[j \cdot \frac{P}{p_i}, (j+1) \cdot \frac{P}{p_i} \right]$, $j = \overline{1, p_i - 1}$ можно разбить исходный диапазон представления чисел *P*, и тем больше вероятность, что два сравниваемых числа не попадут в один интервал и не придется дополнительно вычислять их разность и определять её знак. С другой стороны, чем больше p_i, тем меньше времени необходимо для вычисления l_A, так как от величины p_i зависит размерность чисел l_{A_i}. При этом даже использование $p_i = 2$ в качестве дробящего модуля не гарантирует отсутствия совпадений номеров интервалов для близко лежащих чисел.

Другой подход к сравнению двух чисел в СОК состоит в переходе от СОК к обобщенной полиадической системе счисления (ОПСС). Существует два метода перевода из СОК в ОПСС — традиционный последовательный и модифицированный параллельный. При *последовательном* методе разряды ОПСС вычисляются по формуле

$$a_i \equiv ((...(\alpha_i - a_1)\tau_{1,i} - a_2)\tau_{2,i} - ... - a_{i-1})\tau_{i-1,i} (\mathrm{mod}\, p_i),\ i = \overline{1,n} \qquad (2)$$

где $\tau_{k,j} = \left| \frac{1}{p_k} \right|_{p_j}$ — константы, определяемые набором оснований СОК. При *параллельном* методе перевода необходимо заранее получить ОПСС-представление ортогональных базисов B_i. Далее путем умножения на соответствующий остаток СОК-представления числа производится вычисление ортогональных чисел вида

$$A_i = \left\lfloor 0,0,...,\alpha_i \cdot b_i^i, \alpha_i \cdot b_{i+1}^i, ..., \alpha_i \cdot b_n^i \right\rfloor,\ i = \overline{1,n}. \qquad (3)$$

А само число представляется как сумма его ортогональных чисел $A = \sum_{i=1}^{n} A_i$. При этом может возникнуть переполнение в i-м разряде ОПСС [1, 2]. Для получения конечного результата необходимо выполнить взятие модульных вычетов от каждого разряда ОПСС по соответствующим модулям СОК с последовательным переносом целой части $\lfloor a_i \rfloor_{p_i}$ из младших разрядов в старшие. В последнем разряде переполнение отбрасывается. Последовательный характер операции переноса целой части $\lfloor a_i \rfloor_{p_i}$ является одним из наиболее существенных недостатков данного метода.

Сравнение чисел указанными методами производится в части сравнения их абсолютных величин. Однако, для интервального метода, если номера интервалов исходных чисел *A* и *B* совпадут, то их разность *A*-*B*, как и любой промежуточный результат реальных вычислений в СОК, может оказаться величиной как положительной, так и отрицательной. Поэтому следующей рассматриваемой немодульной операцией в СОК является **определение знака числа**.

Знак числа в СОК можно вводить двумя способами — явно и неявно [1, 2].

При *явном* введении знака числа используется дополнительный бит информации, идентифицирующий принадлежность числа к положительной или отрицательной части множества целых чисел из интервала $(-P; P)$. Тогда операцию вычитания в СОК можно заменить операцией вычитания из числа большего по модулю числа меньшего по модулю. Для данного метода характерно отсутствие временных затрат на вычисление дополнительных параметров, позволяющих определить знак числа, что компенсируется излишеством аппаратной реализации. Но в этом случае мы не сможем без сравнения чисел вычислить их разность, что неприемлемо для интервального метода сравнения.

При *неявном* методе введения знака числа исходный диапазон представления чисел [0, *P*] разбивается на две половины — [0, *P*/2) и [*P*/2, *P*), если среди модулей СОК есть четный, и [0, (*P*-1)/2) и [(*P*-1)/2, *P*), если четного основания нет. Числа принадлежащие первому интервалу считаются положительными, второму — отрицательными. Все действия над числами производятся в дополнительном коде. Дополнительный код положительных чисел совпадает с их СОК-представлением, а для отрицательных чисел получается вычитанием остатков из соответствующих модулей СОК, что эквивалентно числу *P*-*A*. Данный метод обеспечивает простоту реализации операций сложения, вычитания и умножения (за счет выполнения свойства дистрибутивности), но при делении необходимо предварительно восстанавливать СОК-представление отрицательного числа с помощью операции полярного сдвига, то есть вычислять прямой код числа. По завершении деления также необходимо восстанавливать дополнительный код результата для выполнения последующих вычислений.

При неявном введении знака числа для его определения можно использовать интервальный метод сравнения чисел по абсолютной величине с *P*/2 либо с (*P*-1)/2. Альтернативой ему служит метод перевода в ОПСС. При этом если выбрать $p_1 > p_2 > ... > p_n$ и $p_n = 2$, то по значению старшего коэффициента ОПСС-представления (0 или 1) можно определить принадлежность числа к положительной или отрицательной части диапазона представления.

Операции деления и масштабирования в СОК помимо сравнения чисел тесно связаны с **расширением** остаточного представления модулярного кода. Использование округления в процессе масштабирования требует разработки эффективных алгоритмов вычисления округляемого остатка α_{n+1}. Округляемый остаток зависит от величины числа и величины делителя, которые заранее предугадать невозможно. Обозначая делитель через p_{n+1}, получаем формальное расширение диапазона представления чисел $P' = P \cdot p_{n+1}$. На основании этого можно вывести закономерности представления чисел из исходного диапазона $[0, P)$ в новой расширенной СОК с диапазоном $[0, P')$.

Один из способов расширения остаточного представления чисел в СОК — *метод перевода чисел в ОПСС с дополнительным финальным шагом* [2]. Как известно, число в ОПСС представляется в виде

$$A = a_n p_1 p_2 \ldots p_{n-1} + a_{n-1} p_1 p_2 \ldots p_{n-2} + \ldots + a_3 p_1 p_2 + a_2 p_1 + a_1, \quad (4)$$

где p_i – основания ОПСС. Если основания СОК и ОПСС совпадают, то эквивалентны и диапазоны представления чисел. В расширенной ОПСС то же число будет представлено в виде

$$A = a_{n+1} p_1 p_2 \ldots p_n + a_n p_1 p_2 \ldots p_{n-1} + \ldots + a_3 p_1 p_2 + a_2 p_1 + a_1. \quad (5)$$

Приравнивая правые части выражений (4) и (5), получаем, что $a_{n+1} = 0$. На основании формулы (2) получаем выражение для α_{n+1}:

$$0 \equiv ((\ldots(\alpha_{n+1} - a_1)\tau_{1,n+1} - a_2)\tau_{2,n+1} - \ldots - a_n)\tau_{n,n+1} (\mathrm{mod}\, p_{n+1}), \Rightarrow$$

$$\alpha_{n+1} = \left| \left| \sum_{i=1}^{n} a_i \tau_{i,n+1} \right|_{p_{n+1}} \cdot \left| \frac{1}{\left| \prod_{i=1}^{n} \tau_{i,n+1} \right|_{p_{n+1}}} \right|_{p_{n+1}} \right|_{p_{n+1}}, \Rightarrow$$

$$\alpha_{n+1} = \left| \left| \sum_{i=1}^{n} a_i \tau_{i,n+1} \right|_{p_{n+1}} \cdot |P|_{p_{n+1}} \right|_{p_{n+1}} \quad (6)$$

Основным недостатком описанного метода является необходимость вычисления констант $\tau_{i,n+1}$, $i = \overline{1,n}$, как обратных мультипликативных элементов $\left| \frac{1}{p_i} \right|_{p_{n+1}}$. Поэтому данный метод накладывает существенные ограничения на допустимые значения p_{n+1}, связанные с необходимостью существования (вычислимости) обратных мультипликативных элементов, используемых в алгоритме.

Другой способ расширения остаточного представления в СОК состоит в использовании такой позиционной характеристики как *ранг числа* [1, 2]. Согласно методу ортогональных базисов, применяемому для перевода чисел из СОК в ПСС, величина числа определяется по формуле

$$A = \sum_{i=1}^{n} \alpha_i B_i - r_A P, \quad (7)$$

где r_A – *ранг числа* — величина, показывающая во сколько раз в процессе перевода был превышен диапазон представления чисел *P*. Путем взятия модульных вычетов от левой и правой частей (7) можно получить $\alpha_{n+1} \equiv (\sum_{i=1}^{n} \alpha_i B_i - r_A P) \bmod p_{n+1}$ или

$$\alpha_{n+1} \equiv (\left|\alpha_1\beta_1 + \alpha_2\beta_2\right|_{p_{n+1}} + \ldots + \left|\alpha_n\beta_n + r_A\beta_{n+1}\right|_{p_{n+1}}) \bmod p_{n+1}, \quad (8)$$

где $\beta_i = \left|B_i\right|_{p_{n+1}}$, $i = \overline{1,n}$, $\beta_{n+1} = \left|p_{n+1} - \left|P\right|_{p_{n+1}}\right|_{p_{n+1}}$. Алгоритм, представленный формулой (8) для ускорения вычислений можно реализовать методом рекуррентного модульного сдваивания. Расширять остаточное представление, как и в предыдущем случае можно по нескольким модулям параллельно. Главным достоинством описанного метода служит отсутствие в его алгоритме операций вычисления обратных мультипликативных элементов, таких как константы $\tau_{k,j}$, что снимает ряд ограничений на допустимые значения p_{n+1}. Поэтому в отличие от метода на основе перевода в ОПСС, где расширение можно проводить только по модулям взаимно простым с основаниями СОК, данный метод является более универсальным.

Существует еще один метод расширения остаточного представления чисел на основе использования такой позиционной характеристики, как *ядро числа*. По определению ядро числа $R_A = \sum_{i=1}^{n} \tau_i \left[\frac{A}{p_i}\right]$, где τ_i – целые числа. Положив $\tau_1 = \tau_2 = \ldots = \tau_{n-1} = 0$, а $\tau_n = 1$, можно выразить R_A через остатки α_i:

$$R_A = \left(\sum_{i=1}^{n} R_{B_i}\alpha_i\right) \bmod R_D, \quad (9)$$

где $R_{B_i} = B_i / p_n$, $i = \overline{1,n-1}$, $R_{B_n} = (B_n - 1)/p_n$, $R_D = P / p_n = P_{n-1}$. Тогда

$$\alpha_{n+1} \equiv \sigma_1\alpha_1 + \sigma_2\alpha_2 + \ldots + \sigma_n\alpha_n + \theta R_A \pmod{p_{n+1}},$$

где σ_i и θ — коэффициенты, определяемые системой оснований. С учетом предложенного выбора констант τ_i можно записать выражение для α_{n+1} в виде

$$\alpha_{n+1} \equiv \sigma_n\alpha_n + \theta R_A \pmod{p_{n+1}}, \quad (10)$$

где $\sigma_n = \delta_n(1/\varepsilon) \pmod{p_{n+1}}$, $\theta = \pi(1/\varepsilon) \pmod{p_{n+1}}$, $\delta_n = \left|\frac{P}{p_n}\right|_{p_{n+1}}$, $\pi = \left|P\right|_{p_{n+1}}$, $\varepsilon = \left|R_D\right|_{p_{n+1}}$ [1].

Использование формулы (10) для практических вычислений малопригодно, так как в явном виде (10) она требует вычисления обратных мультипликативных элементов $\left|\frac{1}{p_n}\right|_{p_{n+1}}$, что при заранее неизвестном p_{n+1} требует дополнительных временных затрат, а также накладывает дополнительные ограничения на допустимые значения p_{n+1}. Более того, детальный анализ формулы (10) показывает, что путем эквивалентных преобразований она сводится к формуле

$$\alpha_{n+1} \equiv \alpha_n + p_n R_A \pmod{p_{n+1}}, \quad (11)$$

где R_A вычисляется по формуле (9). При желании, вычисления по формуле (9) могут быть реализованы методом рекуррентного модульного сдваивания, за счет чего может быть получен существенный выигрыш в скорости выполнения операций. Основным недостатком формулы (11) является зависимость ядра числа от его остаточного представления, выражаемая формулой (9), то есть необходимость пересчета R_A для каждого числа A. С другой стороны формула (11) не содержит операций вычисления обратных мультипликативных элементов и может быть реализована для любого p_{n+1}.

Сравнивая формулы (8) и (11) можно заметить, что по своей структуре они имеют много общего. Для полноты изложения вопроса о вычислении немодульных характери-

стик СОК, следует рассмотреть ряд методов **вычисления ранга числа** r_A и провести их сравнение по трудоемкости с методом вычисления ядра числа, представленным формулой (9).

Ранг числа изначально появляется в формуле (7), как характеристика, позволяющая определить величину числа в ПСС без редукции накопленной методом ортогональных базисов суммы $\sum_{i=1}^{n}\alpha_i B_i$ по модулю диапазона представления чисел P. Так как r_A входит в формулу (7) в виде произведения $r_A P$, то для выражения r_A из (7) без непосредственного вычисления A необходимо применить *редукцию обеих частей* формулы (7) по некоторому модулю p_{n+1}, удовлетворяющему условию $(P, p_{n+1}) = 1$. Тогда из формулы (7) получаем

$$r_A (\mathrm{mod}\, p_{n+1}) \equiv \left| \frac{\sum_{i=1}^{n}\alpha_i B_i - \alpha_{n+1}}{\overline{P}} \right|_{p_{n+1}}, \qquad (12)$$

где $\alpha_{n+1} = |A|_{p_{n+1}}$. В виду изложенных выше особенностей формулы (7) и полученной на её основе формулы (12) прослеживается тесная взаимосвязь операции вычисления ранга числа и вычисления остатка α_{n+1}, а именно: имея избыточный остаток, мы можем вычислить ранг по формуле (12) и наоборот — имея ранг, мы можем вычислить избыточный остаток по формуле (8).

Существует модификация изложенного метода. Так как ортогональные базисы B_i вычисляются по формуле

$$B_i = m_i P_i = m_i \frac{P}{p_i},\ i = \overline{1,n}, \qquad (13)$$

где $m_i = P_i^{-1} \bmod p_i$ — веса базисов, то (12) можно преобразовать к виду

$$r_A (\mathrm{mod}\, p_{n+1}) \equiv \left| \sum_{i=1}^{n+1}\alpha_i g_i \right|_{p_{n+1}}, \qquad (14)$$

где $g_i = \left| \frac{m_i}{\overline{p_i}} \right|_{p_{n+1}}$, $i = \overline{1,n}$, $g_{n+1} = p_{n+1} - \left| \frac{1}{\overline{P}} \right|_{p_{n+1}}$.

Для реализации еще одного метода вычисления ранга числа необходимо выразить число A формулой (7) в основной (с диапазоном P) и расширенной (с диапазоном P') СОК и приравнять правые части полученных выражений, получим

$$\sum_{i=1}^{n}\alpha_i B_i - r_A P = \sum_{i=1}^{n+1}\alpha_i B_i' - r_A' P' \Rightarrow$$

$$\alpha_{n+1} = \frac{1}{m_{n+1}'}\left(\alpha_1 \frac{m_1 - m_1' p_{n+1}}{p_1} + \alpha_2 \frac{m_2 - m_2' p_{n+1}}{p_2} + \ldots + \alpha_n \frac{m_n - m_n' p_{n+1}}{p_n} - r_A + r_A' p_{n+1} \right) \Rightarrow$$

$$m_{n+1}'\alpha_{n+1} + r_A = -\delta_A + r_A' p_{n+1}, \text{ где } \lambda_i = \frac{m_i' p_{n+1} - m_i}{p_i},\ i = \overline{1,n-1},\ \delta_A = \sum_{i=1}^{n}\alpha_i \lambda_i \text{ [2].}$$

И далее переходя к сравнению по модулю p_{n+1}, получим формулу

$$r_A (\mathrm{mod}\, p_{n+1}) \equiv \left| -\delta_A - m_{n+1}'\alpha_{n+1} \right|_{p_{n+1}}^{+}. \qquad (15)$$

Непосредственные вычисления ранга числа по формулам (12), (14) и (15) сопряжены с рядом практических трудностей:

1. Необходимостью заранее знать остаток α_{n+1} по избыточному модулю p_{n+1}.

2. Необходимостью проводить долгие сложные расчеты промежуточных величин, неизбежно заложенные в алгоритм ($\left|\frac{1}{P}\right|_{p_{n+1}}$ в формуле (12), величины g_i и λ_i в формулах (14) и (15), соответственно).

3. Наличием скрытого ограничения на допустимые значения p_{n+1} (при многократном использовании вычисленного r_A), обозначенного в формулах (12), (14) и (15) операцией редукции по модулю p_{n+1} в левой части сравнения.

Поясняя пункт 3), следует сказать, что в случае если истинное значение ранга числа превысит величину избыточного модуля p_{n+1}, то операция редукции приведет к получению заниженного результата, что может повлечь за собой ошибки расширения остаточного представления числа по формуле (8) для других модулей. Анализ предлагаемых алгоритмов вычисления ранга числа показал, что избыточный модуль p_{n+1} должен быть больше большего основания СОК — при таком подходе вычисленный единожды ранг числа можно использовать для расширения остаточного представления по нескольким избыточным модулям одновременно без дополнительного пересчета r_A по каждому из них. Более того, при изначальном расчете r_A используются операции вычисления обратных мультипликативных элементов, поэтому в этом случае должно выполняться условие $(p_i, p_{n+1}) = 1$, $i = \overline{1,n}$. Несомненно, что при расширении по одному каналу должно выполняться только условие взаимной простоты модулей, так как на конечный остаток α_{n+1} промежуточное сокращение $r_A (\mathrm{mod}\, p_{n+1})$ в данном случае не повлияет. А при дополнительном расширении остаточного представления с заранее корректно вычисленным рангом все дополнительные ограничения на p_{n+1} уже можно будет опустить.

Исходя из вышеизложенных особенностей вычисления r_A, можно сделать вывод, что формулы (12), (14) и (15) малопригодны для практического вычисления и пересчета ранга числа при заранее неизвестной величине избыточного модуля p_{n+1}. На практике для быстрого и качественного расчета r_A представляется удобным всегда иметь один избыточный канал СОК, с величиной модуля p_{n+1}, удовлетворяющей указанным выше требованиям взаимной простоты и допустимой размерности.

Сравнивая формулы для расширения остаточного представления на основе ранга и ядра числа, приходим к выводу, что метод, использующий R_A, выраженный формулой (11), является более удобным для практического применения.

При изложении методов вычисления ранга числа был затронут еще один важный для СОК вопрос — использование избыточных каналов модулярного представления кода наряду с основными. Избыточные каналы в СОК, вообще говоря, могут использоваться не только для вычисления ранга числа, но и для коррекции ошибок, выявления переполнения и выполнения других важных функций.

Коррекция ошибок в СОК это вопрос отдельного комплексного исследования. Существует *геометрический метод* коррекции ошибок СОК-вычислений, базирующийся на том обстоятельстве, что числа в СОК могут принимать определенный разрешенный набор состояний. Множество разрешенных и неразрешенных значений чисел можно изобразить с помощью вершин многогранника в n-мерном пространстве, где n — размерность СОК. *Расстоянием Хэмминга* называется расстояние d между любыми двумя векторами (кодами) a_1 и a_2 из множества P. Под расстоянием в данном случае понимается число компонент, в которых эти векторы (коды) отличаются друг от друга. Согласно известным закономерностям, корректирующий код в СОК может обнаружить все совокуп-

ности из l или меньшего числа ошибок, если минимальное расстояние кода больше l, то есть

$$d_{\min} \geq l+1,$$

и может исправить эти ошибки, если минимальное расстояние кода больше удвоенного числа ошибок, то есть

$$d_{\min} \geq 2l+1.$$

С практической точки зрения исправление ошибок в СОК состоит из трех этапов — обнаружения, локализации и исправления. Возможности данной операции во многом зависят от набора избыточных модулей СОК.

В процессе коррекции модулярного кода может быть установлено, что какие-либо из основных или резервных каналов СОК вышли из строя, что повлечет за собой необходимость динамического перестраивания набора оснований СОК и пересчета связанных с ним констант, используемых в описанных выше алгоритмах. Одними из таких констант являются ортогональные базисы B_i. Рассмотрим *метод пересчета ортогональных базисов* при изменении набора оснований СОК [1, 2]. При изменении набора p_i в формуле (13) изменяются оба множителя, по которым вычисляется B_i. Получаем формулы

а) для *сокращения набора оснований СОК*

$$P/P^* = p_k \Rightarrow m_i^* \equiv m_i p_k (\mathrm{mod}\, p_i) \text{ и } P_i = P_i / p_k \Rightarrow$$

$$B_i^* = \left|B_i\right|_{P^*},\ i = \overline{1,n-1}; \tag{16.1}$$

б) для *расширения набора оснований СОК*

$$P/P^* = 1/p_{n+1} \Rightarrow m_i^* \equiv \frac{m_i}{p_{n+1}} (\mathrm{mod}\, p_i) \text{ и } P_i = P_i p_{n+1} \Rightarrow$$

$$B_i^* = \left|\frac{B_i}{p_{n+1}}\right|_{P^*} p_{n+1},\ i = \overline{1,n}. \tag{16.2}$$

В случае б) дополнительный ортогональный базис B_{n+1} необходимо будет вычислить по общей формуле (13). Предложенный метод пересчета ортогональных базисов позволяет с минимальными затратами динамически пересчитывать значения B_i при изменении набора оснований СОК.

Последняя рассматриваемая немодульная операция в СОК — **обнаружение переполнения**. Переполнение диапазона представления чисел может возникнуть на любом этапе вычислений. В случае если оно не будет вовремя замечено, переполнение способно повлечь за собой последовательное распространение ошибки вычисления во всех расчетах, использующих текущий неверный результат. СОК позволяет разработать эффективный *алгоритм обнаружения переполнения на основе метода перевода в ОПСС* для избыточного набора оснований. Согласно этому алгоритму в упорядоченной ОПСС, то есть при $p_{n+i} > p_j$, $i = \overline{1,r}$, $j = \overline{1,n}$, избыточные цифры ОПСС являются нулевыми — $a_{n+i} = 0$, $i = \overline{1,r}$. Вычисление избыточных цифр ОПСС может проводиться в параллельном режиме и не повлечет больших временных затрат [1, 2]. При этом необходимо заранее знать остаточное представление числа по избыточным модулям.

Таким образом, было проведено теоретическое исследование ряда методов выполнения немодульных операций в СОК и вычисления сопутствующих позиционных характеристик.

Вывод. Операции вычисления ранга и ядра числа, расширения остаточного представления в СОК, вычисления номера интервала и перевода из СОК в ОПСС можно отнести к *классу простых немодульных операций*. На их основе реализуются такие важные операции, как деление, масштабирование, определение знака числа, ошибки и переполнения, которые в свою очередь будем относить к *классу составных немодульных операций* в СОК.

Компьютерное моделирование

Математическая основа всех участвующих в исследовании алгоритмов была изложена выше, и компьютерное моделирование в данном случае призвано подтвердить выдвинутые гипотезы и предположения. С помощью компьютерного эксперимента было вычислено время работы изложенных алгоритмов и получены дополнительные аргументы в защиту сделанных ранее выводов.

Основной упор был сделан на исследовании простых немодульных операций в СОК, как основополагающих для реализации составных операций.

С помощью среды разработки Borland Delphi 2007 была получена программа для расчета позиционных характеристик чисел в СОК и выполнения немодульных операций над ними. В качестве своей основы в программе используется самостоятельно разработанный модуль длинной арифметики для проведения операций над длинными целыми числами в СОК, и многофункциональный динамический список для хранения и обработки наборов длинных целых чисел, моделирующий ячейки памяти СОК-регистров. В структуру указанного динамического списка был включен алгоритм динамического пересчета ортогональных базисов, что носило прикладной характер и позволило на практике проверить его работоспособность, а также была реализована возможность сортировки оснований СОК, что позволило исследовать работоспособность рассмотренных алгоритмов в зависимости от разной последовательности размещения модулей. В программе был реализован высокоточный таймер для вычисления временных затрат.

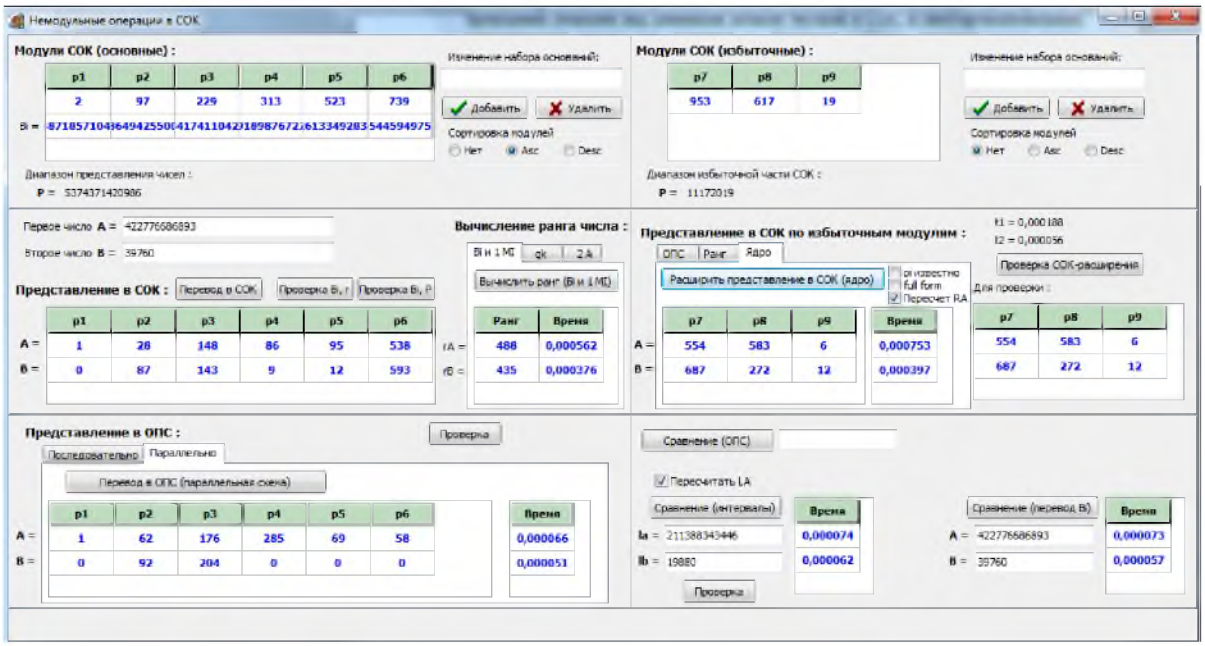

Рис. Вид главного окна программы

В качестве примера набора оснований СОК были выбраны числа $p_1 = 2, p_2 = 97, p_3 = 229, p_4 = 313, p_5 = 523, p_6 = 739$ с рабочим диапазоном представления P = 5374371420986, а в качестве избыточных модулей СОК — числа $p_7 = 953, p_8 = 617, p_9 = 19$. Для интерпретации конечных результатов практического исследования были взяты числа A = 422776686893 и B = 39760.

В процессе предварительных исследований выбирались разные числа и разные наборы модулей, однако для конечного представления результатов специально было подобрано их контрастное сочетание, позволяющее косвенно проследить зависимость временных затрат от размерности получаемых при работе алгоритмов чисел. Использование динамических списков хранения и обработки данных позволило в ряде случаев учитывать не одиночные, а суммарные временные затраты, например, для выполнения задан-

ной операции по всем избыточным модулям СОК, что эквивалентно усреднению вычисляемых временных затрат.

На работу каждого моделируемого алгоритма неизбежно влияют случайные факторы, которые вносят некоторую долю погрешности в тот или иной результат вычисления временных затрат при каждом выполнении заданной операции. Далее приведем один конкретный пример вычисления времени работы, соответствующий среднему наблюдаемому показателю для всех алгоритмов.

Таблица 1

Сравнение временных затрат для методов вычисления ранга числа

Числа	Время работы алгоритмов (с)		
	По формуле (12)	По формуле (14)	По формуле (15)
A	0,000384	0,000069	0,000068
B	0,000377	0,000061	0,000061

Таблица 2

Сравнение временных затрат для методов перевода из СОК в ОПСС

Числа	Время работы алгоритмов (с)	
	Последовательный метод	Параллельный метод
A	0,000171	0,000066
B	0,000136	0,000051

Таблица 3

Сравнение временных затрат для методов расширения остаточного представления

Числа	Время работы алгоритмов (с)				
	Перевод через ОПСС		С использованием ранга числа	С использованием ядра числа	
	с учетом τ	без учета τ		Без пересчета	С пересчетом
A	0,023941	0,000197	0,002001	0,000335	0,000753
B	0,019256	0,000171	0,001964	0,000158	0,000397

Таблица 4

Сравнение временных затрат для методов сравнения чисел

Числа	Время работы алгоритмов (с)	
	Вычисление номера интервала	Восстановление числа методом ортогональных базисов
A	0,000074	0,000073
B	0,000062	0,000057

Результаты сравнения временных затрат для алгоритмов вычисления ранга числа приведены в предположении, что избыточный модуль p_{n+1} и остаток α_{n+1} по нему заранее известны.

Методы перевода из СОК в ОПСС представляют собой отдельный универсальный класс операций, позволяющих одновременно проводить сравнение чисел, определять переполнение и выполнять другие немодульные операции. Поэтому результаты сравнения для этих методов были выделены в отдельную табл. 2. При подсчете времени работы метода параллельного перевода из СОК в ОПСС учитывалась параллельность суммирования ортогональных чисел, и из результатов суммирования по каждому каналу выбирался наиболее продолжительный.

Согласно представленным в табл. 3 результатам подсчета временных затрат для методов расширения остаточного представления в СОК, метод на основе перевода в ОПСС с последующим финальным шагом, при заранее известных константах $\tau_{i,n+1}$, сработал не-

много быстрее аналогичного метода на основе ядра числа. Однако данный метод при учете времени вычисления $\tau_{i,n+1}$ показал увеличение временных затрат на 2 порядка, что явилось наихудшим показателем среди рассмотренных алгоритмов. Результат расширения методом на основе ранга числа был получен без учета дополнительных временных затрат на вычисление r_A. При этом добавление наиболее оптимального показателя по данным таблицы 1 не приводит к существенным изменениям времени работы для данного метода. Совокупные временные затраты для метода расширения остаточного представления в СОК на основе ядра числа приведены в таблице 3 в двух вариантах — без учета времени пересчета R_A для каждого из избыточных модулей и с учетом этого времени. Сравнение совокупного увеличения времени работы алгоритмов на основе ранга и ядра числа при пересчете соответствующих характеристик показывает, что временные затраты на пересчет r_A и R_A имеют одинаковый порядок малости. Однако, итоговое совокупное время работы алгоритма вычисления α_{n+1} на основе ранга числа на порядок выше соответствующего времени работы алгоритма на основе ядра числа.

Таблица 4 содержит временные затраты на вычисление номера интервала числа и на восстановление позиционного представления чисел из СОК методом ортогональных базисов. Эти затраты можно сравнивать как время потраченное на однократное выполнение подготовительного этапа сравнения чисел. Дополнительно данные таблицы 4 можно сравнивать с данными таблицы 2 (в особенности с методом параллельного перевода в ОПСС) по тому же признаку. Оптимальные временные затраты для указанных методов имеют одинаковый порядок малости.

Выводы

1. Использование формул (14) и (15) позволяет на порядок сократить временные затраты на вычисление ранга числа, однако данные методы требуют вычисления и хранения избыточного набора констант g_i и λ_i, соответственно.

2. Метод параллельного перевода в ОПСС на порядок быстрее аналогичного последовательного метода.

3. При практически эквивалентном времени вычисления r_A и R_A в методе на основе ядра числа отсутствуют дополнительные расчеты с использованием всех разрядов СОК-представления, что существенно ускоряет получение конечного результата. Таким образом, метод расширения остаточного представления в СОК на основе ядра числа является наиболее перспективным.

4. Использование алгоритма на основе параллельного перевода в ОПСС более предпочтительно, так как при эквивалентных временных затратах данный метод не только позволяет одновременно выполнять другие функций в СОК, такие, например, как обнаружение переполнения, но и гарантированно не требует проведения дополнительных вычислений, как в случае совпадения номеров интервалов в алгоритме на основе l_A.

Список литературы

1. Червяков Н.И., Сахнюк П.А., Шапошников А.В., Ряднов С.А. Модулярные параллельные вычислительные структуры нейропроцессорных систем / Под. ред. Н.И. Червякова. – М.: ФИЗМАТЛИТ, 2003. – 288 с.

2. Червяков Н.И., Сахнюк П.А., Шапошников А.В., Макоха А.Н. Нейрокомпьютеры в остаточных классах. Кн. 11: Учеб. пособие для вузов. – М.: Радиотехника, 2003. – 272 с.

NOMODAL OPERATIONS RESEARCH IN SYSTEM OF RESIDUAL CLASSES

A.N. LAVRINENKO
N.I. CHERVYAKOV

Stavropol State University

e-mail:
GreatAntonchik-r@yandex.ru
k-fmf-primath@stavsu.ru

The mathematical description some kinds of methods computation positional characteristics for modular code is offered in the article and different aspects of theirs using in nomodal operations executing with numbers in system of residual classes is researched too. The program modeling execution of this algorithm's in computer and allowing to make a comparison of time expenditure's is developed.

Key words: system of residual classes, nomodal operations, computing modeling.