Bull. Nov. Comp. Center, Comp. Science, 19 (2003), 25–38
© 2003 NCC Publisher

Simulation of the diffusion process by the residue number system

V. Markova, D. Jurbin

In this paper, computational potentialities of the residue number system for solution to the diffusion equation is investigated. The finite-difference diffusion scheme is modified in terms of the residue number system. Computer simulation has been performed. The computational characteristics (stability, accuracy) have been assessed and compared with similar ones obtained by the explicit method of solution to the partial differential equation.

Introduction

In recent years, there has been observed a significant shift of numerical computation from general-purpose computers toward parallel computers. To achieve the maximum performance, it is necessary to choose such numerical methods for solution to the partial differential equation (PDE) that would correspond closely to the parallel computer architecture. That is why the explicit methods for the PDE solution are preferred. In such methods the PDE is approximated by a finite difference representation, where time and space are discrete, and a certain physical value is continuous. However, the explicit methods rank below the implicit ones in stability and accuracy. To provide an acceptable stability, the time step should be taken sufficiently small. On the one hand, this brings about an essential increase in computational complexity, and on the other hand – to cumulative round-off errors that make the results fairly unreliable.

As an alternative to the PDE, Cellular Automata (CA) model has been proposed by Toffoli. In the CA model, PDE in floating-point numbers is replaced by Boolean computation in a discrete space. The CA model has some advantages over the finite difference PDE: they are absolutely stable and accurate.

In-between the PDE and the CA models, there are a number of intermediate models, in which both discrete and continuous functions are used. There are gas-lattice with weight connections [1], Lattice–Boltzmann [2], cellular-neural associative networks [3], cellular-neural automaton [4].

In this paper we investigate the potentialities of the residue number system for the diffusion equation. The Residue Number System (RNS) [5–8] belongs to an unweighted NS. The basis for any RNS is a set of relatively prime integers (moduli). The RNS uniquely presents any integer as a set of remainders with respect to each modulo (the RNS representation). As distinct from a weighted NS, all remainders (digits) are independent. The length of each remainder is smaller than that of an initial integer. These two properties of the RNS representation provide parallel, carry-free, high-speed arithmetic (addition, subtraction and multiplication). In addition, the RNS arithmetic is exact (without overflow) and therefore free of round-off error. However, the RNS is found to be inferior to the binary number system in conventional computation since the sign detection, and division are slowly.

In this paper, a finite difference scheme of the diffusion equation, where time, space, and a certain physical value are discrete, is represented in the RNS. To obtain an exact realization of division in a finite difference scheme by the RNS, two strategies are used: transfer of a remainder to the next iteration and representation of the diffusion coefficient as a fraction. (This scheme is further referred to as the RNS scheme.) Moreover, a simple division algorithm is proposed. Computer simulation has shown that the time complexity of our algorithm is less by half as opposed to the well-known algorithm from [5].

Numerical simulation has been performed. The computational characteristics (stability, accuracy) have been investigated. When investigating, a PDE solution in the floating-point numbers is used for comparison.

Experimental results show that the RNS scheme is stable over a wide range of values for an inverse diffusion number as opposed to the conventional scheme used in the PDE. The RNS scheme provides an acceptable accuracy of solution.

The paper is organized as follows. Section 1 describes the main operations in the RNS. The RNS scheme for simulation of the diffusion equation and the results of investigations are given in Section 2.

1. The Residue Number System Arithmetic

1.1. The RNS representation

Let $\mathcal{P} = \{p_0, p_1, \dots, p_{k-1}\}$ be a set of pairwise relatively prime integers, such that $\mathrm{GCD}(p_i, p_j) = 1$ for $i \neq j$ and $\mathrm{GCD}(p_i, p_j)$ denote the greatest common divisor of p_i and p_j. The set $\mathcal{P}$ is called *the moduli set.* The interval $[0, M)$, $M = \prod_{i=0}^{k-1} p_i$, determines *the dynamic range* of the system. Then any integer $X \in [0, M)$ has a unique *RNS representation* [5–8] given by

$$X \to (\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_{k-1},) = \boldsymbol{X},$$

where $\boldsymbol{x}_i = X \bmod p_i$ is *the i-th remainder* of X modulo p_i. The remainder $\boldsymbol{x}_i$ is calculated in the following way

$$X \bmod p_i = \boldsymbol{x}_i = X - \lfloor X/p_i \rfloor p_i,$$

where $\lfloor Y \rfloor$ denotes the largest integer smaller or equal to Y.

As distinct from the weighted NS, the RNS representation has two properties:

- all remainders are independent,
- the length of each remainder is smaller than that of an initial integer.

These two properties of the RNS representation provide parallel, carry-free, high-speed arithmetic.

The weighted-to-RNS conversion is reduced to division of an integer by each modulus from the set $\mathcal{P}$. It is obvious, that the division of large integers can be a slow and impracticable procedure for high-speed calculations. Here we used the algorithm from [8], which is based on the well-known "divide and conquer" technique.

Let $\mathcal{P} = \{p_0, p_1, \dots, p_{k-1}\}$ be a moduli set and $k = 2^t$. If $k < 2^t$, then the moduli equal to 1 are added to the set $\mathcal{P}$. First, the algorithm calculates $k/2$ moduli as the products of two initial moduli p_0p_1, p_2p_3, $\dots$, p_kp_{k-1}. Then, the algorithm calculates $k/4$ moduli as products of the two moduli obtained $p_0p_1p_2p_3$, $p_4p_5p_6p_7$, $\dots$ and so on. This procedure lasts until two moduli $P^1 = p_0p_1 \dots p_{k/2-1}$ and $P^2 = p_{k/2}p_{k/2+1} \dots p_{k-1}$ are obtained.

Further, the remainders are calculated using "divide and conquer" technique. First, the algorithm finds the remainders $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ of the integer X modulo P^1 and modulo P^2, respectively. As a result, the initial task of the size of k is reduced to two tasks of the size of $k/2$. This technique is recursively applied until all the remainders of the integer X are calculated.

Example 1. Let $\mathcal{P} = \{3, 5, 7, 11\}$, $X = 289$. Then $P^1 = p_0p_1 = 15$, $P^2 = p_2p_3 = 77$. According to the above algorithm, firstly, we get two remainders: $\boldsymbol{y}_1 = X \bmod P^1 = 4$ and $\boldsymbol{y}_2 = X \bmod P^2 = 58$. Secondly, we calculate four remainders: $\boldsymbol{x}_0 = \boldsymbol{y}_1 \bmod p_0 = 1$, $\boldsymbol{x}_1 = \boldsymbol{y}_1 \bmod p_1 = 4$, $\boldsymbol{x}_2 = \boldsymbol{y}_2 \bmod p_2 = 2$, and $\boldsymbol{x}_3 = \boldsymbol{y}_2 \bmod p_3 = 3$. In result, $\boldsymbol{X} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = (1, 4, 2, 3)$.

The RNS-to-weighted conversion is carried out by the Chinese algorithm with data preprocessing [8]. The algorithm is based on the Chinese Remainder Theorem [8], which asserts the following.

Let $\mathcal{P} = \{p_0, p_1, \dots, p_{k-1}\}$ be a set of pairwise relatively prime moduli. Let $\boldsymbol{X} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_{k-1})$ be a remainder set. Then

$$X = \sum_{i=0}^{k-1} M_i d_i \boldsymbol{x}_i \bmod M, \tag{1}$$

where $M_i = M/p_i$ is a product of all the moduli, except p_i, $d_i = M_i^{-1} \bmod p_i$ is the multiplicative inverse of M_i modulo p_i.

Similarly to the weighted-to-RNS conversion algorithm, for calculating (1) the Chinese algorithm is based on the same technique. In addition, a set of the multiplicative inverse of M_i modulo p_i is known in advance.

At first, the Chinese algorithm calculates the products of the form

$$q_{ij} = \prod_{m=0}^{i+2^j-1} p_m, \tag{2}$$

where $i = 0, 1, \dots, k-1$, i is multiple of 2^j, $j = 1, 2, \dots, t$.

Further, the integers

$$s_{ij} = \prod_{m=0}^{i+2^j-1} \frac{q_{ij} d_m \boldsymbol{x}_m}{p_m}$$

are determined. If $j = 0$, $0 \le i \le k$, then $s_{i0} = d_i \boldsymbol{x}_i$. If $j = 1, 2, \dots, t$, then s_{ij} is recursively formed

$$s_{ij} = s_{i,j-1} q_{i+2^j,j-1} + s_{i+2^{j-1},j-1} q_{i,j-1} \tag{3}$$

The integer $s_{0t} = X$.

Example 2. Let $\mathcal{P} = \{3, 5, 7, 11\}$, $M = 1155$, $\boldsymbol{X} = (2, 4, 4, 3)$, $d_1 = 1$, $d_2 = 1$, $d_3 = 2$, $d_4 = 2$.

According to (2), we get $q_{00} = p_0 = 3$, $q_{10} = p_1 = 5$, $q_{20} = p_2 = 7$, $q_{30} = p_3 = 11$, $q_{01} = p_0 p_1 = 15$, $q_{21} = p_2 p_3 = 77$, $q_{02} = p_0 p_1 p_2 p_3 = 1155$.

For $i = 0, 2$, $j = 1$ we have $s_{00} = d_0 \boldsymbol{x}_0 = 2$, $s_{10} = d_1 \boldsymbol{x}_1 = 4$, $s_{20} = d_2 \boldsymbol{x}_2 = 8$, $s_{30} = d_3 \boldsymbol{x}_3 = 8$.

Further, s_{ij} is obtained according to (3).

For $i = 0, 2$, $j = 1$ we get $s_{01} = s_{00} q_{10} + s_{10} q_{00} = 22$, $s_{21} = s_{20} q_{30} + s_{30} q_{20} = 130$.

For $i = 0$, $j = 2$ we have $X = s_{02} \bmod 1155 = (s_{01} q_{21} + s_{21} q_{01}) \bmod 1155 = (2277 + 13015) \bmod 1155 = 179$.

It is known [5–8], that one of the most important considerations, when designing the RNS systems, is the choice of the moduli set (the form as well as the number of the moduli chosen). The moduli set significantly affects the dynamic range length, the speed of the RNS processing, the complexity of the weighted-to-RNS and, especially, the RNS-to-weighted conversion.

1.2. The RNS arithmetic

Let $\mathcal{P} = \{p_0, p_1, \dots, p_{k-1}\}$ be a set of pairwise relatively prime moduli located in increasing order (this condition is necessary only for the division given below). Let $\boldsymbol{X} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_{k-1})$ and $\boldsymbol{Y} = (\boldsymbol{y}_0, \boldsymbol{y}_1, \dots, \boldsymbol{y}_{k-1})$ be two RNS representations of the integers X and Y, $X, Y \in [0, M)$. Then the RNS representation of the integer $Z = X \oplus Y$, $Z \in [0, M)$, is given by

$$\boldsymbol{X} \oplus \boldsymbol{Y} = \boldsymbol{Z} = (\boldsymbol{z}_0, \boldsymbol{z}_1, \dots, \boldsymbol{z}_{k-1}), \tag{4}$$

where $\oplus$ denotes addition, subtraction or multiplication, $\boldsymbol{z}_i = (\boldsymbol{x}_i \oplus \boldsymbol{y}_i) \bmod p_i$ for all $i = 0, 1, \dots, k-1$.

Example 3. Let $\mathcal{P} = \{3, 5, 7, 11\}$, $\boldsymbol{X} = (2, 3, 5, 9)$, $\boldsymbol{Y} = (1, 1, 2, 0)$. Then according to (4), we have $\boldsymbol{X} + \boldsymbol{Y} = (3, 4, 0, 9)$, $\boldsymbol{X} - \boldsymbol{Y} = (1, 1, 4, 9)$, $\boldsymbol{X} \cdot \boldsymbol{Y} = (2, 3, 3, 0)$. These results can be easily checked up.

Equation (4) demonstrates the parallel, carry-free nature of the Residue Number System. In addition, the RNS arithmetic is exact and therefore free of round-off error. However, such operations as sign detection, division are difficult in the RNS. Here we consider a special case of division: division by module.

Division. Let $\mathcal{P} = (p_0, p_1, \dots, p_{k-1})$, $\boldsymbol{X} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_{k-1})$ be the RNS dividend, $\boldsymbol{p}_i = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \dots, \boldsymbol{\pi}_{i-1}, 0, \underbrace{p_i, \dots, p_i}_{k-1-i})$ be the RNS devisor. If the number X is divided by the module p_i, then $\boldsymbol{x}_i = 0$, otherwise, the number $X' = X - \boldsymbol{x}_i$ is used as dividend. Then the division

$$\frac{\boldsymbol{X}}{\boldsymbol{p}_i} = \left(\frac{\boldsymbol{x}_0}{\boldsymbol{\pi}_0}, \frac{\boldsymbol{x}_1}{\boldsymbol{\pi}_1}, \dots, \frac{\boldsymbol{x}_{i-1}}{\boldsymbol{\pi}_{i-1}}, \frac{0}{0}, \frac{\boldsymbol{x}_{i+1}}{p_i}, \dots, \frac{\boldsymbol{x}_k}{p_i}\right) = (\boldsymbol{z}_0, \boldsymbol{z}_1, \dots, \boldsymbol{z}_{k-1}) = \boldsymbol{Z}$$

is carried out in two steps.

At the first step, the algorithm generates the first approximation of the quotient

$$\hat{\boldsymbol{Z}} = (\boldsymbol{z}_0, \boldsymbol{z}_1, \dots, \boldsymbol{z}_{i-1}, 0, \boldsymbol{z}_{i+1}, \dots, \boldsymbol{z}_{k-1}),$$

where the digits $\boldsymbol{z}_j$, $j \neq i$, are calculated by division of the digit $\boldsymbol{x}_j$ by an appropriate digit of the divider ($\boldsymbol{\pi}_j$ or p_i). To avoid uncertainty, the digit $\boldsymbol{z}_i$ is equated to zero. If the digit $\boldsymbol{x}_j$ is not divisible by $\boldsymbol{\pi}_j$ or p_i, then division is replaced by multiplication

$$\frac{\boldsymbol{x}_j}{\boldsymbol{\pi}_j} \bmod p_j \to \left(\boldsymbol{x}_j \cdot \frac{1}{\boldsymbol{\pi}_j} \bmod p_j\right) \bmod p_j,$$

where $\frac{1}{\boldsymbol{\pi}_j}$ is the multiplicative inverse of $\boldsymbol{\pi}_j$ modulo p_j. For the given moduli, the multiplicative inverse values are formed in advance. For example, $\frac{\boldsymbol{x}}{5} \bmod 11 = \boldsymbol{x} \cdot \frac{1}{5} \bmod 11 = (\boldsymbol{x} \cdot 9) \bmod 11$. (The multiplicative inverse values are shown in Table 1.)

Table 1

p	Multiplicative inverse values for π									
	1	2	3	4	5	6	7	8	9	10
2	1									
3	1	2								
5	1	3	2	4						
7	1	4	5	2	3	6				
11	1	6	4	3	9	2	8	7	5	10

At the second step, the algorithm determines the digit value $\boldsymbol{z}_i$. Here we present a very simple method for calculating the digit value $\boldsymbol{z}_i$. Computer simulation has shown that the time complexity of our method is half as large as the method from [5].

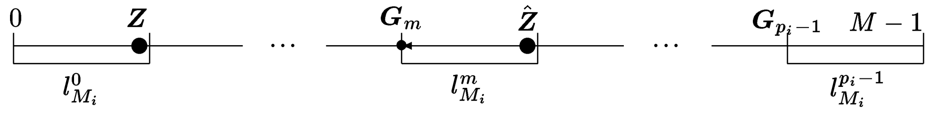

Figure 1. Graphical representation of the interval $[0, M)$

So, the first approximation of the quotient $\hat{\boldsymbol{Z}}$ can belong to one of the intervals $l_{M_i}^m$, $M_i = M/p_i$, $m = 0, \dots, p_i - 1$, in the range $[0, M)$ (Figure 1). These intervals are derived from splitting the range to p_i parts. Each interval $l_{M_i}^m$ contains M_i numbers. The first number of the m-th interval takes the form

$$\boldsymbol{G}_m = (0, 0, \dots, 0, \underset{i}{m}, 0, \dots, 0).$$

Here $\boldsymbol{g}_{mj} = 0$ for all $j \neq i$ because the number G_m is a multiple of each module p_j. As far as $Z \leq M/p_i$, the quotient $\boldsymbol{Z}$ always belongs to the interval $[0, M_i)$.

It is obvious that both numbers $\boldsymbol{Z}$ and $\hat{\boldsymbol{Z}}$ are at the same distance from the beginnings of the appropriate intervals, i.e.,

$$\boldsymbol{Z} - \mathbf{0} = \hat{\boldsymbol{Z}} - \boldsymbol{G}_m. \tag{5}$$

For the i-th remainder equation (5) can be written down as

$$\boldsymbol{z}_i - 0 = (0 - m) \bmod p_i.$$

Hence, $\boldsymbol{z}_i = (0 - m) \bmod p_i$.

So, to assess the number of the interval, in which the first approximation of the quotient $\hat{\boldsymbol{Z}}$ is located, it is necessary to determine $\boldsymbol{G}_m$ according to $\hat{\boldsymbol{Z}}$. Such a determination is reduced to a successive increase of the number of zero remainders in the RNS representation of the integer (initial or intermediate). This process is further called *nullivization.*

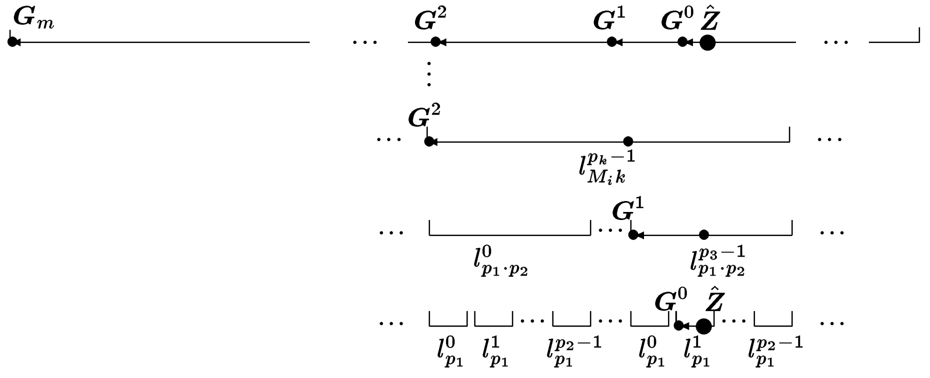

Figure 2. Graphical representation of nullivization

The nullivization is produced in $(k-1)$ steps. It begins from the least nonzero remainder except the i-th remainder (the modulo p_i is the devisor). The process can be represented as follows:

$$\hat{\boldsymbol{Z}} \to (0, \boldsymbol{g}_1^1, \dots, \boldsymbol{g}_{k-1}^1) = \boldsymbol{G}^0 \to (0, 0, \boldsymbol{g}_2^2, \dots, \boldsymbol{g}_{k-1}^2) = \boldsymbol{G}^1 \to \dots \to \boldsymbol{G}_m.$$

Here the moduli p_0, p_1 are not dividers. The number $\boldsymbol{G}^0$ is the first one of the least interval, in which the number $\hat{\boldsymbol{Z}}$ is located (Figure 2). The least interval l_{p_0} consists of p_0 numbers: p_1 intervals l_{p_0} form the interval $l_{p_0 p_1}$, in which the number $\boldsymbol{G}^1$ is the first one, etc. Thus, nullivization can be interpreted as a sequential move from the number $\hat{\boldsymbol{Z}}$ in the beginnings of intervals, starting with the least one, until the number $\boldsymbol{G}_m$ is attained.

Now we will explain the technique of nullivization. It can be represented as iterative procedure. Let $\boldsymbol{G}^{j-1} = (0, \dots, 0, \boldsymbol{g}_j^{j-1}, \dots, \boldsymbol{g}_{k-1}^{j-1})$, $j < i$, be the result of the $(j-1)$th step of nullivization. Then

$$\boldsymbol{G}^j = \begin{cases} \boldsymbol{G}^{j-1}, & \text{if } \boldsymbol{g}_j^{j-1} = 0 \text{ for all } j \neq i, \\ \boldsymbol{G}^{j-1} - \boldsymbol{M}^{p_j}_{\boldsymbol{g}_j^{j-1}} & \text{otherwise,} \end{cases}$$

where the number $\boldsymbol{M}^{p_j}_{\boldsymbol{g}_j^{j-1}} = (0, \dots, 0, \boldsymbol{g}_j^{j-1}, \dots, \boldsymbol{g}_{k-1}^{j})$, is called *a nullivization constant.* For modulo p_j, the number of the nullivization constants equal to $p_j - 1$. A nullivization constant should be multiple of the interval length in the beginning of which we have moved at the $(j-1)$th step. Otherwise the residues, being set into zero at the previous steps, cease to be in zero. As a rule, the nullivization constants are calculated in advance.

Example 4. Let $\mathcal{P} = \{3, 5, 7, 11\}$, $X = 305$, p_2 be the devisor. The quotient $\boldsymbol{Z} = (\boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3) = \frac{(2,0,4,8)}{(2,0,5,5)}$ is calculated as follows.

First, we obtain the remainders of the quotient $\hat{\boldsymbol{Z}}$, except $\hat{\boldsymbol{z}}_1$:

$$\hat{\boldsymbol{z}}_0 = \frac{2}{2} \bmod 3 = \left(2 \cdot \frac{1}{2} \bmod 3\right) \bmod 3 = (2 \cdot 2) \bmod 3 = 1,$$
$$\hat{\boldsymbol{z}}_2 = \frac{4}{5} \bmod 7 = \left(4 \cdot \frac{1}{5} \bmod 7\right) \bmod 7 = (4 \cdot 3) \bmod 7 = 5,$$
$$\hat{\boldsymbol{z}}_3 = \frac{8}{5} \bmod 11 = \left(8 \cdot \frac{1}{5} \bmod 11\right) \bmod 11 = (8 \cdot 9) \bmod 11 = 6.$$

So, $\hat{\boldsymbol{Z}} = (1,0,5,6)$ ($\hat{Z}_{10} = 985$). For the set $\mathcal{P} = \{3,5,7,11\}$ the nullivization constants are tabulated in Table 2. The nullivization process is shown in Figure 3. In order that $\boldsymbol{g}_0^0 = 0$, we choose the constant M_1^3 since $\hat{\boldsymbol{z}}_0 = 1$. As a result we have the integer $\boldsymbol{G}^0 = \hat{\boldsymbol{Z}} - \boldsymbol{M}_1^3 = (0,4,4,5)$. Similarly, the constant

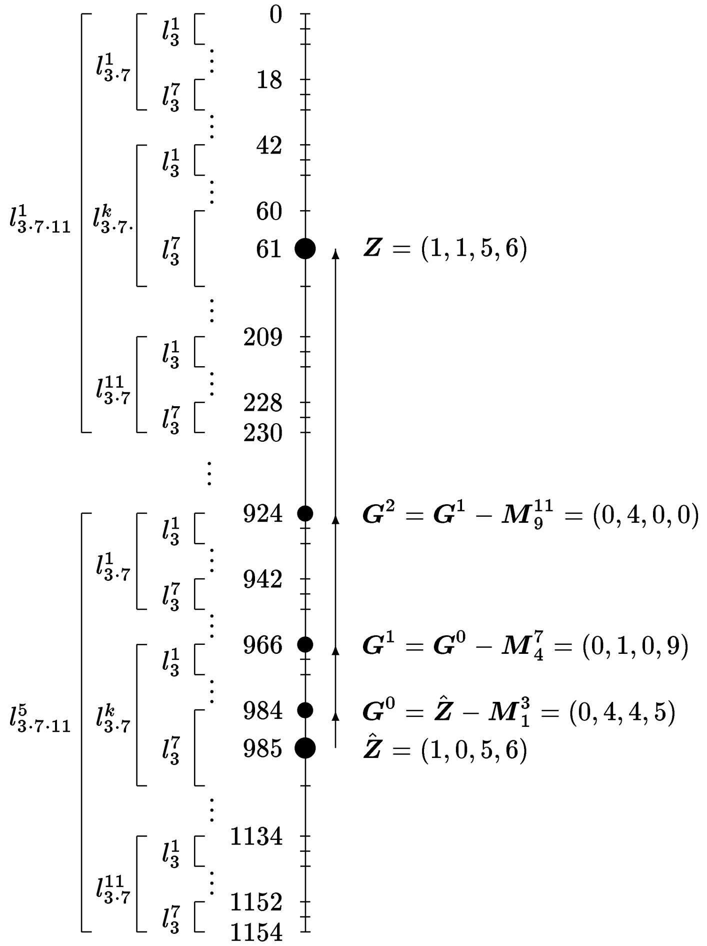
$l^1_{3\cdot7\cdot11}$
$l^1_{3\cdot7}$
$l^k_{3\cdot7\cdot}$
$l^{11}_{3\cdot7}$
l^1_3
l^7_3
0
18
42
60
61
209
228
230
924
942
966
984
985
1134
1152
1154
$l^5_{3\cdot7\cdot11}$
$l^k_{3\cdot7}$
$\boldsymbol{Z} = (1,1,5,6)$
$\boldsymbol{G}^2 = \boldsymbol{G}^1 - \boldsymbol{M}_9^{11} = (0,4,0,0)$
$\boldsymbol{G}^1 = \boldsymbol{G}^0 - \boldsymbol{M}_4^7 = (0,1,0,9)$
$\boldsymbol{G}^0 = \hat{\boldsymbol{Z}} - \boldsymbol{M}_1^3 = (0,4,4,5)$
$\hat{\boldsymbol{Z}} = (1,0,5,6)$

Figure 3. Example of division

Table 2

$p_1 = 3$	$p_3 = 7$	$p_4 = 11$
$\boldsymbol{M}_1^3 = (1,1,1,1)$	$\boldsymbol{M}_1^7 = (0,0,1,4)$	$\boldsymbol{M}_1^{11} = (0,0,0,1)$
$\boldsymbol{M}_2^3 = (2,2,2,2)$	$\boldsymbol{M}_2^7 = (0,4,2,9)$	$\boldsymbol{M}_2^{11} = (0,4,2,0)$
	$\boldsymbol{M}_3^7 = (0,3,3,3)$	$\boldsymbol{M}_3^{11} = (0,3,0,3)$
	$\boldsymbol{M}_4^7 = (0,3,4,7)$	$\boldsymbol{M}_4^{11} = (0,2,0,4)$
	$\boldsymbol{M}_5^7 = (0,2,5,1)$	$\boldsymbol{M}_5^{11} = (0,1,0,5)$
	$\boldsymbol{M}_6^7 = (0,1,6,6)$	$\boldsymbol{M}_6^{11} = (0,0,0,6)$
		$\boldsymbol{M}_7^{11} = (0,4,0,7)$
		$\boldsymbol{M}_8^{11} = (0,3,0,8)$
		$\boldsymbol{M}_9^{11} = (0,2,0,9)$
		$\boldsymbol{M}_{10}^{11} = (0,1,0,10)$

M_4^7 sets up the second remainder into "zero". Further, the number $\boldsymbol{G}_m$ is calculated by subtracting M_9^{11} from the obtained number $\boldsymbol{G}^2 = (0,1,0,9)$.

So, the quotient $\boldsymbol{Z}$ belongs to the 4-th interval, that is, $4 \cdot 231 \leq (1,0,5,6) < 5 \cdot 231$. Hence, $\boldsymbol{z}_2 = (0-4) \bmod 5 = 1$ and $\boldsymbol{Z} = (1,1,5,6)$ $(Z = 65)$.

2. Simulation of diffusion by Residue Number System

2.1. Finite difference diffusion scheme

It is customary the diffusion equation is represented as follows:

$$\frac{\partial u}{\partial t'} = \alpha \frac{\partial^2 u}{\partial x^2}, \tag{6}$$

where u is temperature, α is the diffusion coefficient, x is a coordinate of $1D$ continuous space. When solving this equation by a finite difference scheme, space and time are discretized, so that $x = ih$, $t' = t\tau$, where i, t being integers, h and τ – the space and the time steps. After such a discretization, equation (6) takes the following form

$$\frac{u_i^{t+1} - u_i^t}{\tau} = \alpha \frac{u_{i-1}^t + u_{i+1}^t - 2u_i^t}{h^2}, \tag{7}$$

where $t = 0, 1, \ldots$, u_i^t, $i = 0, 1, \ldots, N$, is a value of the integer function u at nodes in the lattice. Initial conditions are given as integers.

In the iterative form equation (7) is written down as

$$u_i^{t+1} = u_i^t + \frac{1}{D}(u_{i-1}^t + u_{i+1}^t - 2u_i^t) = u_i^t + \frac{1}{D}L(u_i^t), \tag{8}$$

where $L(u_i^t) = u_{i-1}^t + u_{i+1}^t - 2u_i^t$, $\frac{1}{D} = \frac{\alpha\tau}{h^2}$. The value $\frac{1}{D}$ is called *the inverse diffusion coefficient.*

2.2. Implementation of finite difference scheme by the RNS

Scheme (8) is very simple, but some difficulties arise when carrying out the division by the RNS. To obtain an exact realization of division in a finite difference scheme, we use the following two strategies.

Transfer of a remainder to the next iteration. Let us consider the division in equation (8). Let $t = 0$, and $L(u_i^0) \bmod D \neq 0$. Then we represent $L(u_i^0)$ as $\tilde{L}(u_i^0) + R_i^1$, where $\tilde{L}(u_i^0) = \lfloor L(u_i^0)/D \rfloor D$, R_i^1 is the remainder, and transfer the resulting remainder to the 1st iteration. Such a strategy eliminates the loss of accuracy.

Then the finite difference scheme (8) takes the following form

$$u_i^{t+1} = u_i^t + \left\lfloor \frac{L(u_i^t) + R_i^{t+1}}{D} \right\rfloor, \tag{9}$$

where R_i^{t+1} is recursively calculated

$$R_i^{t+1} = (L(u_i^t) + R_i^t) \bmod D, \quad R_i^0 = 0. \tag{10}$$

Representation of the diffusion coefficient as fraction. Since the number $1/D$ is a fraction, by definition, then we represent the number D as ratio between the integers $D = D_2/D_1$. Such a replacement transforms the recurrences (9), (10) into

$$u_i^{t+1} = u_i^t + \left\lfloor \frac{D_1 L(u_i^t) + R_i^t}{D_2} \right\rfloor, \tag{11}$$

$$R_i^{t+1} = [D_1 L(u_i^t) + R_i^t] \bmod D_2, \quad R_i^0 = 0. \tag{12}$$

Bellow, scheme (11), (12) for diffusion solution is referred to as *the RNS scheme* as opposed to scheme (8) in *the floating-point numbers* (*FP scheme*).

In this paper (Section 1.1), we have defined the RNS for non-negative numbers $x \in [0, M)$. While calculating the numerator in (11) it may be $P_i^t = D_1 L(u_i^t) + R_i^t < 0$. If it is so, then (11) is to be modified as follows

$$u_i^{t+1} = u_i^t - \left\lfloor \frac{\hat{P}_i^t}{D_2} \right\rfloor,$$

where $\hat{P}_i^t = M - P_i^t$. Here the sign of the value $P_i^t = D_1 L(u_i^t) + R_i^t$ is determined by the algorithm [9].

2.3. Computer simulation results

Here we investigate the following computational characteristics of the diffusion RNS scheme: stability, accuracy. Numerical modeling has been performed on Pentuim III. The experiment has been done for the RNS with $\mathcal{P} = (5, 7, 11, 13)$. Moreover, two functions (smooth and discontinuous), were used as initial distribution. Both functions are determined on $[0, 90)$ and have the following form

$$f_1(x) = \begin{cases} 1000\cos(x) + 1000, & 0 \le x < 40, \\ 0, & 40 \le x < 90; \end{cases} \qquad f_2(x) = \begin{cases} 2000, & 0 \le x < 10, \\ 500, & 10 \le x < 30, \\ 1500, & 30 \le x < 40, \\ 0, & 40 \le x < 90. \end{cases}$$

The RNS scheme stability is understood as the ability of a scheme to quench oscillations in a solution and to provide their non-occurrence. It is well-known [10], that the FP scheme is stable if $D \ge 2$.

Here the stability investigation is reduced to an experimental estimation of the value of the inverse diffusion coefficient D such that the RNS scheme is stable for the two initial distributions. For this purpose, the dependencies the dispersion on time were obtained for different values of D.

Let $u|_{t=0} = f_1$. Experiments show that the RNS scheme is stable if $D > 1.64$. Figure 4 shows the time dependence of dispersion both in the RNS and the FP schemes for $D = 1.64$. Here both schemes are instable. As distinct from the RNS scheme, an instability of the FP scheme grows faster. As is seen from Figure 4, the dispersion curve of the FP solution lies below the dispersion curve of the RNS solution. Two solutions of diffusion equation for $D = 1.65$ are presented in Figure 5. In this case the RNS scheme is stable while the FP scheme is unstable. Moreover, the oscillation amplitude is in exceed of the function values at the 28-th step.

Let $u|_{t=0} = f_2$. The experiments show that the RNS scheme is stable if $D \ge 1.98$. So, the condition of the RNS scheme stability with the discon-

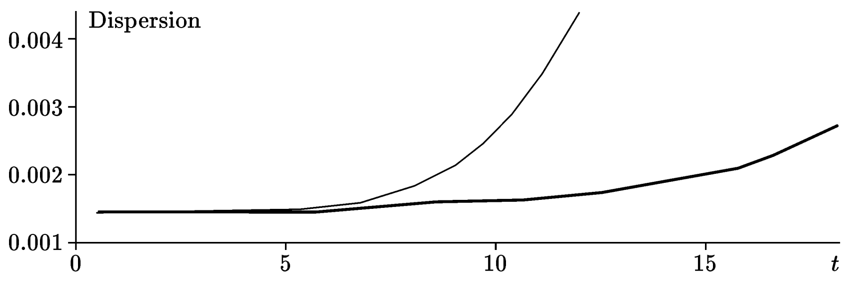

Figure 4. The time dependence of of the dispersion in the RNS (thick line) and the FP (thin line) solutions with $u|_{t=0} = f_1$ and $D = 1.64$

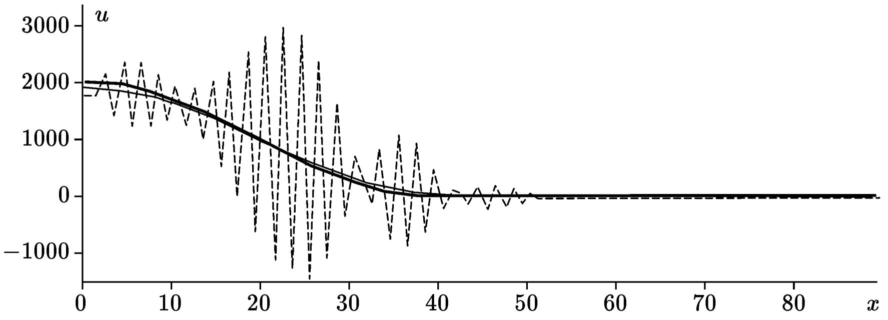

Figure 5. The RNS (thin line) and the FP (dashed line) solutions of the diffusion equation with $t = 28$, $D = 1.65$, $u|_{t=0} = f_1$ in comparison with the initial distribution $f_1(x)$ (thick line)

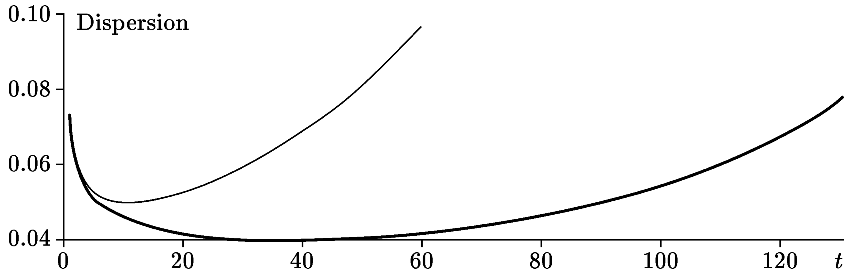

Figure 6. The time dependence of the oscillation intensity in the RNS (thick line) and the FP (thin line) solutions with with $u|_{t=0} = f_2$ and $D = 1.97$

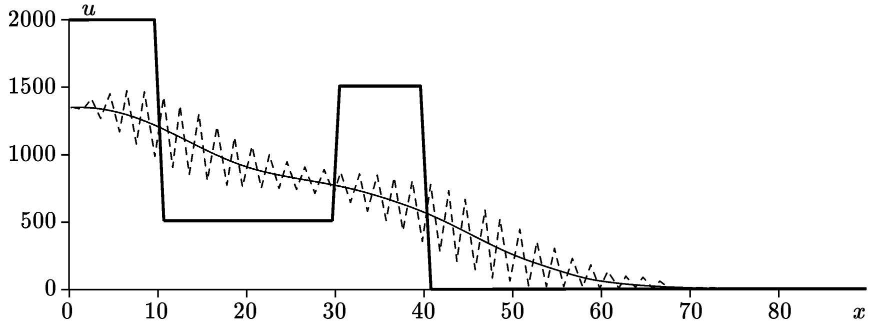

Figure 7. The RNS (thin line) and the FP (dashed line) solutions of the diffusion equation with $t = 135$, $D = 1.98$, $u|_{t=0} = f_2$ in comparison with the initial distribution $f_2(x)$ (thick line)

tinuous initial distribution differs little from the condition of the FP scheme stability. Figure 6 shows the time dependence of the dispersion both in the RNS and the FP schemes for two values of D. For $D = 1.97$ both schemes are unstable. Similarly, experimental results with the smooth initial distribution, the oscillations in the PF solution grow significantly faster. For $D = 1.98$ (Figure 7) the RNS scheme is stable, the FP scheme is unstable.

The simulation has shown that the RNS scheme is stable over a wide range of values for the inverse diffusion coefficient D as opposed to the FP scheme. Moreover, the value of D depends on the initial distribution.

Accuracy of the RNS solution. Obviously, for a given set of moduli RNS, an acceptable accuracy of this solution is provided by the absence of round-off errors and the RNS scheme.

Here when studying the accuracy, the FP solution is considered to be standard for comparison. Experiments have shown that the RNS solution and the FP solution coincide very closely for smooth and discontinuous initial distributions. To estimate the degree of their agreement, a function $\Delta u = u_{\mathrm{RNS}} - u_{\mathrm{FP}}$ (difference of two solutions) is introduced. From Figures 8 and 9 follows that a relative error is about 0.05–0.1%.

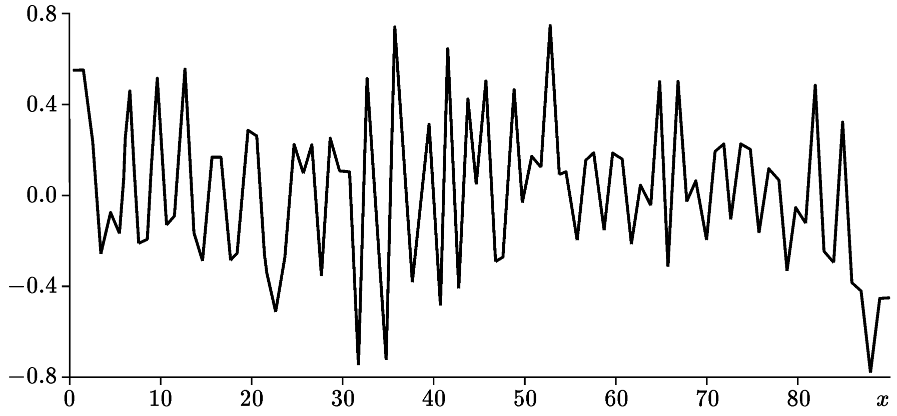

Figure 8. Difference of solutions with $t = 1000$, $D = 2.00$, $u|_{t=0} = f_1$

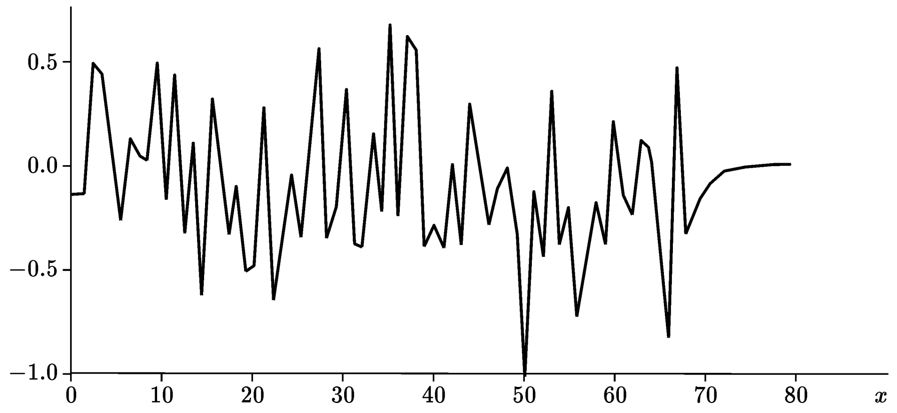

Figure 9. Difference of solutions with $t = 1000$ steps, $D = 3.00$, $u|_{t=0} = f_2$

3. Conclusion

In this paper, the RNS potentialities for differential equations solution have been studied. Simulation results enable us to make the following conclusions:

- The RNS scheme is stable over a wide range of values for an inverse diffusion number as opposed to the well-established scheme used in the PDE.
- The RNS scheme provides an acceptable accuracy of the solution.

As expected, the RNS solution with implementing in general purpose computers has lower performance than the PDE solution. Even realization of the RNS solution using OpenMP enables to increase the performance less than 15%. Improvement of the time complexity can be achieved due to the designing specialized computing devices, which would support parallelism both on the remainder and the bit levels.

References

[1] Rothman D.H., Zaleski S. Lattice-Gas Cellular Automata. Simple Models of Complex Hydrodynamics. – Cambridge: University Press, 1997.

[2] Chen S., Wang Z., Shan X., Doolen G.D. Lattice-Boltzmann computational fluid dynamics in three dimensions // J. Statistical Physics. – 1992. – Vol. 68, № 3/4. – P. 379–407.

[3] Bandman O. Discrete-continuous models for spatial dynamic simulations // NCC Bulletin, Series Computer Science. – Novosibirsk: NCC Publisher, 2002. – Issue 17. – P. 17–29.

[4] Bandman O. Cellular-neural automaton: a Hybrid model for reaction-diffusion simulation // Future Generation Computer Systems. – 2002. – № 18. – P. 737–745.

[5] Akyshskii I.J., Udizkii D.I. Computer Arithmetic in Residue Number System. – Moscow: Sov. Radio, 1968.

[6] Torgashov V.A. Residue Number System and Computer. – Moscow: Sov. Radio, 1973.

[7] Aho A., Hopcroft J., Ullman J. The Design and Analysis of Computer Algorithms. – Moscow: Mir, 1979.

[8] Taylor F.J. Residue arithmetic: a tutorial with examples // IEEE Comput. Mag. – 1984. – Vol. 17, №. 5. – P. 50–62.

[9] Vu T.V. Efficient implementation of the Chinese remainder theorem for sign detection and residue decoding // IEEE Trans. Comput. – 1985. – Vol. 34, № 7. – P. 646–651.

[10] Roache Patrick J. Computational Fluid Dynamics. – Moscow: Mir, 1980.