PROCEEDINGS OF SCIENCE

Error-Free Algorithms to Solve Special and General Discrete Systems of Linear Equations

Miroslav Morháč[1]
Institute of Physics, Slovak Academy of Sciences
Dúbravska cesta 9, Bratislava, 845 11, Slovak republic
E-mail:Miroslav.Morhac@savba.sk

Abstract

The paper presents a survey of error-free algorithms to solve various systems of linear equations. The presented algorithms do not introduce computational errors into the solution and thus they are well suited to solve ill-conditioned linear systems. The error-free algorithms are based on modulo arithmetic. Two basic approaches have been investigated in the paper. The first one is based on iterative scheme using one modulus only. The other one is parallel and uses several moduli and the Chinese theorem. It is based on polynomial algebra operations that allow to express the operation of deconvolution as a sequence of convolutions of both response and output signals.

XI International Workshop on Advanced Computing and Analysis Techniques in Physics Research
Amsterdam, the Netherlands
23-27 April, 2007

[1] Speaker

PoS(ACAT)073

1. Introduction

The solution of linear algebraic equations is a very frequent task in numerical mathematics and experimental physics at all. When solving such a set of linear equations one often meets the problem of ill-conditioned matrix of the set. For large dense sets of linear equations, which are as a rule ill-conditioned, the stability of the solution cannot be guaranteed. Rounding-off and truncation errors, during the numerical computations, involved in obtaining the solution to the problem cannot be tolerated. Let us illustrate the situation using a simple example.

Illustrative example:

Let us have ill-conditioned set of two linear equations

$$\begin{aligned} x + 3y &= 4 \\ x + 3.00001y &= 4.00001, \end{aligned} \tag{1}$$

which has the solution $x=1, y=1$. On the other hand let us consider the set

$$\begin{aligned} x + 3y &= 4 \\ x + 2.99999y &= 4.00002. \end{aligned} \tag{2}$$

This set has the solution $x = 10, y = -2.$ Very small change in the coefficients (0.00002 and 0.00001) caused enormous changes in the solution. The inverse matrix of (1) contains elements of the order of 10^5. It demonstrates its ill-conditionality.

We are motivated by the fact that in scientific computations there are large classes of ill-conditioned problems and there are also numerically unstable algorithms. Digital computer is a finite machine and therefore it is capable of representing internally only a finite set of numbers. There exist difficulties associated with the attempts at approximating arithmetic in the field of real numbers $(R,+,-)$ by using the finite set of so-called floating-point numbers F, more appropriately called the set of computer-representable numbers. There is no possibility of representing the continuum of real numbers in any detail. A "practical" solution is to represent a real number by the closest computer-representable number, thereby introducing the rounding error.

It is well known that computers can perform certain arithmetic operations exactly if the operands are integers. Moreover there are many situations in practice when we work with integer input data. Processing of spectra (histograms) in nuclear physics can serve as a good example. When solving ill-conditioned problems during the analysis of spectra it is not reasonable to leave the world of exact integers and thus introduce instability. It motivated us to consider integer arithmetic as a mean of avoiding rounding errors in the hope that certain ill-conditioned problems can be solved exactly. Residue number system is an example of a number system with which we can do exact arithmetic. The aim of the contribution is to present error-free algorithms to solve ill-conditioned systems of linear equations.

Let us summarize how we can proceed when the solution of ill-conditioned linear system of equations can be expressed as rational fractions of integers

- we can increase the precision of floating point representation - however it does not need to guarantee the correct result

- we can work with long integers - from algorithmic point of view it is many times very cumbersome
- we can work with integers in finite rings using residue class or modulo arithmetic and at the end of the calculation to convert the modulo representation into real numbers.

In the rest of the paper when speaking about modulo arithmetic we shall consider prime moduli. Let us now illustrate modulo representation of negative numbers and inverse numbers for modulus $M = 17$ (see Table I and II).

x	0	1	2	3	4	5	6	7	8	-8	-7	-6	-5	-4	-3	-2	-1
$x(\mathrm{mod}\, M)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Table I. Modulo representation of numbers from the range $\langle -(M-1)/2, (M-1)/2 \rangle$ for $M = 17$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$x^{-1}(\mathrm{mod}\, M)$	0	1	9	6	13	7	3	5	15	2	12	14	10	4	11	8	16

Table II. Modulo inverse numbers for $M = 17$

One can observe that the assignment between numbers and their negatives and inverses is prime moduli uniquely unique. By definition for 0 we have taken inverse number to be 0.

2. Convolution systems

In many applications the dynamic behavior of a linear system can be described by means of impulse response function $h(n)$. The periodical convolution of finite sets $x(n)$ and $h(n)$ is

$$y(n) = \sum_{m=0}^{N-1} h(n-m)x(m), \quad n = 0,1,\ldots,N-1, \tag{3}$$

where N is the length of both vectors. Very frequently the advantageous property of factorization of the convolution of two signals to the product of their Fourier coefficients is utilized [1], [2]

$$F_i\left[y(n)\right] = F_i\left[h(n)\right] \cdot F_i\left[x(n)\right], \quad i = 0,1,\ldots,N-1. \tag{4}$$

Since the Fourier transform cannot be computed with absolute precision, the number theoretical transforms were defined with the aim to carry out the fast error-free convolution of data. These transforms are defined in the ring of integers using the operations carried out in modulo M arithmetic, where M is prime. Direct and inverse Number Theoretical Transforms (NNT) can be defined

$$X(k) = \left[\sum_{n=0}^{N-1} x(n)\alpha^{kn}\right] (\mathrm{mod}\, M) \quad k = 0,1,\ldots,N-1, \tag{5}$$

$$x(n) = \left[N^{-1}\sum_{k=0}^{N-1} X(k)\alpha^{-kn}\right] (\mathrm{mod}\, M) \quad n = 0,1,\ldots,N-1, \tag{6}$$

where

a. $N \cdot N^{-1} (\mathrm{mod}\, M) = 1,$

b. $\alpha^N \,(\mathrm{mod}\, M) = 1,$

and N must divide $M-1$. Among NTT the most famous are Merssene and Fermat transforms [2]. They are frequently used for error-free calculations of convolutions. However, the problem of precision is much more critical in the inverse operation, i.e., deconvolution.

2.1 Precise deconvolution using the Fermat number transform [3]

Convolution system (3) can be written in matrix form

$$\begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-2) \\ y(N-1) \end{bmatrix} = \begin{bmatrix} h(0), & h(N-1), & \cdots & h(2), & h(1) \\ h(1), & h(0), & \cdots & h(3), & h(2) \\ \vdots & \vdots & & \vdots & \vdots \\ h(N-2), & h(N-3), & \cdots & h(0), & h(N-1) \\ h(N-1), & h(N-2), & \cdots & h(1), & h(0) \end{bmatrix} \bullet \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-2) \\ x(N-1) \end{bmatrix} \tag{7}$$

or

$$\mathbf{y} = H \cdot \mathbf{x}. \tag{8}$$

The solution of any regular system of linear equations (8) can be expressed as

$$\mathbf{x} = \frac{1}{D}\left[M^0 \mathbf{x}_0 + M^1 \mathbf{x}_1 + M^2 \mathbf{x}_2 \cdots + M^m \mathbf{x}_m \right], \tag{9}$$

where D is determinant of the matrix H, M is chosen modulus (in our case Fermat number) and m is a finite number. Substituting (9) into matrix equation (8) gives

$$\mathbf{y} = \frac{1}{D}\left[M^0 \cdot H \cdot \mathbf{x}_0 + M^1 \cdot H \cdot \mathbf{x}_1 + \cdots + M^m \cdot H \cdot \mathbf{x}_m \right] \tag{10}$$

or after some modifications

$$\frac{1}{M}\left[\cdots \frac{1}{M}\left[\frac{1}{M}\left(\mathbf{y} \cdot D - H \cdot \mathbf{x}_0 \right) - H \cdot \mathbf{x}_1 \right] - \cdots H \cdot \mathbf{x}_m \right] = \mathbf{0}. \tag{11}$$

From (11) we can define the vectors

$$\mathbf{y}_1 = \frac{1}{M}\left(\mathbf{y} \cdot D - H \cdot \mathbf{x}_0 \right), \quad \mathbf{y}_{j+1} = \frac{1}{M}\left(\mathbf{y}_j - H \cdot \mathbf{x}_j \right), \quad j = 1,2,\ldots,m. \tag{12}$$

If the vector $\mathbf{y}_{j+1}$ equals the zero vector the calculation can be finished. It can serve as an indication of the end of iterations.

To calculate modulo solution of (8) we can utilize the Fermat number transform. For transformed values it holds

$$R_i(\mathbf{x}) = R_i^{-1}(\mathbf{h}) \cdot R_i(\mathbf{y}) (mod\ M) \quad i = 0,1,\ldots,N-1. \tag{13}$$

Using inverse Fermat transform we obtain modulo M solution of the vector $\mathbf{x}$

$$\mathbf{x}(mod\ M) = \mathbf{x_M} = \frac{D}{D}\mathbf{x_M}(mod\ M) = \frac{D_M}{D}\mathbf{x_M}(mod\ M), \tag{14}$$

where

$$D = k \cdot M + D_M, \mathbf{x}_0 = D_M \cdot \mathbf{x}_M, \quad D_M = \left[\prod_{i=0}^{N-1} H(i) \right] (mod\ M). \tag{15}$$

Then

$$\mathbf{x}_0 = \mathbf{x} \cdot D - M\left(\mathbf{x}_1 + M\left(\mathbf{x}_2 \ldots + M\mathbf{x}_m\right)\ldots\right). \tag{16}$$

Substituting (16) to (12) yields

$$\mathbf{y}_1 = H \cdot \mathbf{x}_1 + M\left(H \cdot \mathbf{x}_2 + \ldots M \cdot H \cdot \mathbf{x}_m\right)$$

or in general

$$\mathbf{y}_j (mod\, M) = H \cdot \mathbf{x}_j, \quad j = 1,2,\ldots,m. \tag{17}$$

Hence to find particular solution $\mathbf{x}_j$ we can apply again Fermat transform. Then we can proceed to the next iteration step

$$\mathbf{y}_{j+1} = \frac{\mathbf{y}_j - H \cdot \mathbf{x}_j}{M}. \tag{18}$$

The final solution $\mathbf{x}$ can be calculated using (9).

In (9), (10), (11), (12), (14) we assumed we know the exact value of the determinant D of the convolution matrix H. In what follows we introduce the algorithm to calculate it [3], [4]. By employing the Fourier transform the determinant of the convolution matrix of the system (8) can be expressed

$$D = (-1)^{\frac{1}{2}(N-1)(N-2)+1} \cdot \prod_{k=0}^{N-1}\left(h_0 + h_1 W^k + h_2 W^{2k} + \cdots + h_{N-1} W^{(N-1)k}\right) \tag{19}$$

where $W = e^{\frac{-j\cdot(2\pi)}{N}}$ and N is a power of 2, i.e., the determinant of such a matrix can be expressed as the product of the coefficients of the Fourier transform of the response vector $\mathbf{h}$. The Fourier transform is used only formally. For details we refer to [3].

Let us illustrate the algorithm by an example with $N = 8$. Applying DFT to vector $\mathbf{h}$ we obtain the bit reversed vector

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & W^2 & -1 & -W^2 & 1 & W^2 & -1 & -W^2 \\ 1 & -W^2 & -1 & W^2 & 1 & -W^2 & -1 & W^2 \\ 1 & W & W^2 & W^3 & -1 & -W & -W^2 & -W^3 \\ 1 & -W & W^2 & -W^3 & -1 & W & -W^2 & W^3 \\ 1 & W^3 & -W^2 & W & -1 & -W^3 & W^2 & -W \\ 1 & -W^3 & -W^2 & -W & -1 & W^3 & W^2 & W \end{bmatrix} \cdot \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \end{bmatrix} =$$

$$= \begin{bmatrix} h_0 + h_1 + h_2 + h_3 + h_4 + h_5 + h_6 + h_7 \\ h_0 - h_1 + h_2 - h_3 + h_4 - h_5 + h_6 - h_7 \\ (h_0 - h_2 + h_4 - h_6) + (h_1 - h_3 + h_5 - h_7)W \\ (h_0 - h_2 + h_4 - h_6) - (h_1 - h_3 + h_5 - h_7)W \\ (h_0 - h_4) + (h_1 - h_5)W + (h_2 - h_6)W^2 + (h_3 - h_7)W^3 \\ (h_0 - h_4) - (h_1 - h_5)W + (h_2 - h_6)W^2 - (h_3 - h_7)W^3 \\ (h_0 - h_4) + (h_1 - h_5)W^3 - (h_2 - h_6)W^2 + (h_3 - h_7)W \\ (h_0 - h_4) - (h_1 - h_5)W^3 - (h_2 - h_6)W^2 - (h_3 - h_7)W \end{bmatrix} = \begin{bmatrix} a_0 \\ b_0 \\ c_0 + c_1 W^2 \\ c_0 - c_1 W^2 \\ d_0 + d_1 W + d_2 W^2 + d_3 W^3 \\ d_0 - d_1 W + d_2 W^2 - d_3 W^3 \\ d_0 + d_1 W^3 - d_2 W^2 + d_3 W^3 \\ d_0 - d_1 W^3 - d_2 W^2 - d_3 W^3 \end{bmatrix} = \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} \begin{matrix} \} 1.\ \textbf{group} \\ \} 2.\ \textbf{group} \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} 3.\ \textbf{group} \\ \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} 4.\ \textbf{group} \end{matrix}$$

The numbers in the first and the second group are real. In the third group we multiply the numbers X_2, X_3 and in the fourth group the numbers X_4, X_5, X_6, X_7. The fact that $W^4 = -1$ must be taken into account. Then

$$X_2 \cdot X_3 = \left(c_0 + c_1 W^2\right)\left(c_0 - c_1 W^2\right) = c_0^2 + c_1^2$$

$$X_4 \cdot X_5 = \left(d_0 + d_1 W + d_2 W^2 + d_3 W^3\right)\left(d_0 - d_1 W + d_2 W^2 - d_3 W^3\right) =$$

$$= \left(d_0^2 + 2d_1 d_3 - d_2^2\right) - \left(d_1^2 + 2d_0 d_2 - d_3^2\right) W^2 = e_0 - e_1 W^2$$

$$X_6 \cdot X_7 = \left(d_0 + d_1 W^3 - d_2 W^2 + d_3 W\right)\left(d_0 - d_1 W^3 - d_2 W^2 - d_3 W\right) =$$

$$= \left(d_0^2 + 2d_1 d_3 - d_2^2\right) + \left(d_1^2 + 2d_0 d_2 - d_3^2\right) W^2 = e_0 + e_2 W^2$$

$$X_4 \cdot X_5 \cdot X_6 \cdot X_7 = \left(e_0 - e_1 W^2\right)\left(e_0 + e_1 W^2\right) = e_0^2 + e_1^2.$$

We see that both products are real numbers.

Generally, let us have the vectors ${}^0\mathbf{X}_n, {}^1\mathbf{X}_n, {}^2\mathbf{X}_n, \ldots$, where for ${}^k\mathbf{X}_n$ and ${}^{k+1}\mathbf{X}_n$ it holds

$${}^k\mathbf{X}_n = \begin{bmatrix} {}^kX_n(j-1) \\ \vdots \\ {}^kX_n(j/2) \\ {}^kX_n(j/2-1) \\ \vdots \\ {}^kX_n(1) \\ {}^kX_n(0) \end{bmatrix}, \qquad {}^{k+1}\mathbf{X}_n = \begin{bmatrix} -{}^kX_n(0) \\ {}^kX_n(j-1) \\ \vdots \\ {}^kX_n(j/2+1) \\ {}^kX_n(j/2) \\ \vdots \\ {}^kX_n(2) \\ {}^kX_n(1) \end{bmatrix},$$

$n = 0,1,\ldots,\log_2 N - 1$ is the number of reduction, $k = 0,1,\ldots,N/2^{n+1}$ is the number of cyclic shifts with a negative transition to the topmost position and $j = N/2^n$. Then for the reduced vector ${}^0\mathbf{X}_{n+1}$ we get

$${}^0\mathbf{X}_{n+1} = \begin{bmatrix} \vdots \\ (-1)^k \left[{}^kX_n^2(0) - {}^kX_n^2(j/2) + 2\sum_{i=1}^{\frac{j}{2}-1} (-1)^i \cdot {}^kX_n(i+1) \cdot {}^kX_n(j-i-1) \right] \\ \vdots \\ (-1)^0 \left[{}^0X_n^2(0) - {}^0X_n^2(j/2) + 2\sum_{i=1}^{\frac{j}{2}-1} (-1)^i \cdot {}^0X_n(i+1) \cdot {}^0X_n(j-i-1) \right] \end{bmatrix}.$$

Repeating the above described procedure until the length of the vector ${}^0\mathbf{X}_{n+1}$ is greater than 1, we can calculate the exact integer value of the product of the appropriate group. Multiplying the products of all groups we can get the determinant of the convolution matrix, which is used in the above outlined deconvolution algorithm.

Illustrative example:

Let us have the convolution system

$$\begin{bmatrix} 3 & 0 & 0 & 2 \\ 2 & 3 & 0 & 0 \\ 0 & 2 & 3 & 0 \\ 0 & 0 & 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 3 \\ 0 \end{bmatrix}, \tag{20}$$

where the determinant $D = 65.$ We shall use the modulus $M = 17$, which is the Fermat number and $N = 4.$ Then the forward and inverse Fermat transform matrices are as follows

$$T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & 16 & 13 \\ 1 & 16 & 1 & 16 \\ 1 & 13 & 16 & 4 \end{bmatrix}, \qquad T^{-1} = 13 \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 13 & 16 & 4 \\ 1 & 16 & 1 & 16 \\ 1 & 4 & 16 & 13 \end{bmatrix}.$$

Then

$$\mathbf{h_t} = T \cdot \mathbf{h}(mod\ 17) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & 16 & 13 \\ 1 & 16 & 1 & 16 \\ 1 & 13 & 16 & 4 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 2 \\ 0 \\ 0 \end{bmatrix} (mod\ 17) = \begin{bmatrix} 5 \\ 11 \\ 1 \\ 12 \end{bmatrix}.$$

Employing Euclidian algorithm we calculate vector of inverse modulo values

$$\mathbf{h_t}^{-1} = \left[5^{-1}, 11^{-1}, 1^{-1}, 12^{-1}\right]^T (mod\ 17) = [7,14,1,10]^T.$$

1-st iteration step

We calculate the vector of the transformed output values

$$\mathbf{Y_{m0}} = T \cdot \mathbf{y_{m0}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & 16 & 13 \\ 1 & 16 & 1 & 16 \\ 1 & 13 & 16 & 4 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 5 \\ 3 \\ 0 \end{bmatrix} (mod\ 17) = \begin{bmatrix} 11 \\ 3 \\ 1 \\ 14 \end{bmatrix}.$$

Then due to factorization property of the Fermat number transform (13) we get

$$\mathbf{X}_0 = \begin{bmatrix} 11 & \cdot & 7 \\ 3 & \cdot & 14 \\ 1 & \cdot & 1 \\ 14 & \cdot & 10 \end{bmatrix} (mod\ 17) = \begin{bmatrix} 9 \\ 8 \\ 1 \\ 4 \end{bmatrix}.$$

Applying the inverse Fermat transform we obtain

$$\mathbf{x}_0 = 13 \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 13 & 16 & 4 \\ 1 & 16 & 1 & 16 \\ 1 & 4 & 16 & 13 \end{bmatrix} \cdot \begin{bmatrix} 9 \\ 8 \\ 1 \\ 4 \end{bmatrix} (mod\ 17) = \begin{bmatrix} 14 \\ 15 \\ 8 \\ 6 \end{bmatrix} = \frac{D}{D} \begin{bmatrix} 14 \\ 15 \\ 8 \\ 6 \end{bmatrix} = \frac{D_M}{D} \begin{bmatrix} 14 \\ 15 \\ 8 \\ 6 \end{bmatrix} (mod\ 17) = \frac{1}{65} \begin{bmatrix} 9 \\ 6 \\ 10 \\ 16 \end{bmatrix},$$

where D_M was calculated using relation (15). Let us suppose that the positive numbers are represented in the range $\langle 0,(M-1)/2 \rangle$ and the negative ones in the range $\langle (M-1)/2+1, M-1 \rangle$. Then we obtain the negative values for numbers greater than $(M-1)/2$ by subtracting the modulus

$$\mathbf{x}_0 = \frac{1}{65}\left[-8,6,-7,-1\right]^T .$$

Further we calculate

$$\mathbf{y}_0{}' = H \cdot \mathbf{x}_0 = \frac{1}{65}\begin{bmatrix} 3 & 0 & 0 & 2 \\ 2 & 3 & 0 & 0 \\ 0 & 2 & 3 & 0 \\ 0 & 0 & 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} -8 \\ 6 \\ -7 \\ -1 \end{bmatrix} = \frac{1}{65}\begin{bmatrix} -26 \\ 2 \\ -9 \\ -17 \end{bmatrix},$$

$$\mathbf{y}_1 = \frac{1}{M}\left(\mathbf{y_{m0}} \cdot D - \mathbf{y}_0{}'\right) = \frac{1}{17}\left[\begin{bmatrix} 195 \\ 325 \\ 195 \\ 0 \end{bmatrix} - \begin{bmatrix} -26 \\ 2 \\ -9 \\ -17 \end{bmatrix}\right] = \begin{bmatrix} 13 \\ 19 \\ 12 \\ 1 \end{bmatrix},$$

$$\mathbf{y_{m1}} = \mathbf{y_1}(mod\,17) == \left[13,2,12,1\right]^T .$$

2-nd iteration step

In the next iteration step we obtain

$$\mathbf{Y_{m1}} = T \cdot \mathbf{y_{m1}}(mod\,17) = \left[11,5,5,14\right]^T , \quad \mathbf{X}_1 = \left[11 \cdot 7,5 \cdot 14,5 \cdot 1,14 \cdot 10\right]^T (mod\,17) = \left[9,2,5,4\right]^T .$$

Using the inverse Fermat transform we calculate the vector

$$\mathbf{x}_1 = T^{-1} \cdot \mathbf{X}_1(mod\,17) = \left[5,3,2,16\right]^T .$$

Adjusting the elements greater than $(M-1)/2$ we obtain $\mathbf{x}_1 = \left[5,3,2,-1\right]^T$. Then we calculate

$$\mathbf{y}_1{}' = H \cdot \mathbf{x}_1 = \left[13,19,12,1\right]^T , \quad \mathbf{y}_2 = \frac{\mathbf{y}_1 - \mathbf{y}_1{}'}{M} = \frac{1}{17}\left[\left[13,19,12,1\right]^T - \left[13,19,12,1\right]^T\right] = \left[0,0,0,0\right]^T .$$

The zero vector $\mathbf{y}_2$ indicates the end of calculation. The resulting vector is then obtained by

$$\mathbf{x} = \frac{1}{D}\left[\mathbf{x}_0 + \mathbf{x}_1 \cdot 17\right] = \frac{1}{65}\left[\left[-8,6,-7,-1\right]^T + 17 \cdot \left[5,3,2,-1\right]^T\right] = \frac{1}{65}\left[77,57,27,-18\right]^T .$$

This is the exact solution of our illustrative example (20).

The periodical k-dimensional convolution of the finite sets $x(n_1,n_2,\ldots,n_k)$ and $h(n_1,n_2,\ldots,n_k)$ is defined by

$$y(n_1,n_2,...,n_k) = \sum_{m_1=0}^{N-1}\sum_{m_2=0}^{N-1}\ldots\sum_{m_k=0}^{N-1} x(m_1,m_2,...m_k) \cdot h(n_1-m_1,n_2-m_2,...,n_k-m_k).$$

Analogously to one-dimensional case in [5] we have derived the error-free algorithm of multidimensional deconvolution as well as algorithm to calculate the determinant of the multidimensional convolution matrix.

2.2 Error-free deconvolution using polynomial algebra concept [6], [7]

Polynomial algebra plays an important role in digital signal processing because convolutions can be expressed in terms of operations on polynomials [2]. Polynomial algebra is the basis of one group of fast and error-free one- and multidimensional convolution algorithms.

Let us suppose the elements of $h(m), x(l)$ to be coefficients of the polynomials $H(z)$ and $X(z)$ of the degree $N-1$. Hence we have

$$H(z)=\sum_{m=0}^{N-1} h(m) z^m, \quad X(z)=\sum_{l=0}^{N-1} x(l) z^l. \tag{21}$$

If we multiply $H(z)$ by $X(z)$, the resulting polynomial will be of degree $2N-2$. Thus

$$Y(z)=H(z)\cdot X(z)=\sum_{n=0}^{2N-2} y(n) z^n. \tag{22}$$

This means that the convolution of two sequences can be treated as the product of two polynomials.

If the one-dimensional convolution defined by (1) is cyclic the indices n,m,l are calculated modulo N. This implies that $z^N (mod\ N)=1$ and therefore the cyclic convolution can be considered as the multiplication of two polynomials

$$Y(z)=H(z)\cdot X(z), \tag{23}$$

where the powers of the polynomial variable z are calculated modulo N or by

$$Y(z)=H(z)\cdot X(z) mod\left(z^N-1\right). \tag{24}$$

Let us proceed to multidimensional case of convolution. The k-dimensional cyclic convolution of the discrete input signal x and the response signal h is defined

$$y(n_1,\ldots n_k)=\sum_{m_1=0}^{N_1-1},\ldots,\sum_{m_k=0}^{N_k-1} h(m_1,\ldots m_k)\, x(n_1-m_1,\ldots,n_k-m_k), \tag{25}$$

where $n_1\in\langle 0,N_1-1\rangle$, $n_2\in\langle 0,N_2-1\rangle$, …, $n_k\in\langle 0,N_k-1\rangle$ and indices n_1-m_1, are calculated modulo N_1, indices n_2-m_2 are calculated modulo N_2, etc. By considerations analogous to those of the one-dimensional case the polynomial expression of the k-dimensional convolution is obtained

$$Y(z_1,z_2,\ldots,z_k)=H(z_1,z_2,\ldots,z_k)\cdot X(z_1,z_2,\ldots,z_k), \tag{26}$$

where the powers of the variable z_1 are calculated modulo N_1 and the powers of the variable z_k are calculated modulo N_k. In the case of deconvolution, i.e. when we know the output signal y and the response signal h, the input signal can be formally expressed in polynomial form

$$X(z)=Y(z)\cdot H^{-1}(z), \tag{27}$$

and

$$X(z_1,z_2,\ldots,z_k)=Y(z_1,z_2,\ldots,z_k)\cdot H^{-1}(z_1,z_2,\ldots,z_k), \tag{28}$$

for one-, and k-dimensional deconvolution, respectively. Let us illustrate the algorithm of calculation of (27) and (28) by the following examples.

Illustrative example for one-dimensional deconvolution:

Let the vectors of output signal and impulse response be

$y=[3,1,2,1]^T$, $h=[1,4,2,0]^T$.

According to (23) it holds

$$(3+z+2z^2+z^3)=(1+4z+2z^2)\cdot X(z).$$

Formally we can express the sought

$$X(z)=\frac{3+z+2z^2+z^3}{1+4z+2z^2}.$$

Multiplying the numerator and the denominator by the polynomial $H(-z)(mod\ z^N-1)$ we obtain

$$X(z)=\frac{(3+z+2z^2+z^3)(1-4z+2z^2)}{(1+4z+2z^2)(1-4z+2z^2)}=\frac{(3-9z+4z^2-5z^3)}{(5-12z^2)}.$$

In the following sections of the paper the product $H(z)\cdot H(-z)$ will be called the reduction of the polynomial $H(z)$ with respect to the variable z. If we multiply the numerator and the denominator by the polynomial $5+12z^2$, the resulting polynomial is

$$X(z)=\frac{(3-9z+4z^2-5z^3)(5+12z^2)}{(5-12z^2)(5+12z^2)}=\frac{63-105z+56z^2-133z^3}{-119}.$$

The coefficients of the polynomial $X(z)$ represent exact solution of the example.

Illustrative example for two-dimensional deconvolution:

Let the matrices of two-dimensional output signal and two-dimensional impulse response be

$$y=\begin{bmatrix}3, & 2\\ 1, & 4\end{bmatrix},\quad h=\begin{bmatrix}2, & 3\\ 1, & 3\end{bmatrix}.$$

Using (26) for k = 2 we have

$$[(3+z_1)+(2+4z_1)\cdot z_2]=[(2+z_1)+(3+3z_1)\cdot z_2]\cdot X(z_1,z_2)$$

or

$$X(z_1,z_2)=\frac{[(3+z_1)+(2+4z_1)\cdot z_2]}{[(2+z_1)+(3+3z_1)\cdot z_2]}.$$

By multiplying the numerator and the denominator by the polynomial $H(z_1,-z_2)$, , i.e., by its reduction with respect to the variable z_2, we get

$$X(z_1,z_2)=\frac{[(3+z_1)+(2+4z_1)\cdot z_2][(2+z_1)-(3+3z_1)\cdot z_2]}{[(2+z_1)+(3+3z_1)\cdot z_2][(2+z_1)-(3+3z_1)\cdot z_2]}=\frac{(11+13z_1)+(4+2z_1)z_2}{13+14z_1}.$$

Now we multiply this result by the polynomial $13-14z_1$

$$X(z_1,z_2)=\frac{[(11+13z_1)+(4+2z_1)z_2](13-14z_1)}{(13+14z_1)(13-14z_1)}=\frac{(39-15z_1)+(-24+30z_1)z_2}{27}.$$

The resulting matrix is

$$x = \frac{1}{27}\begin{bmatrix} 39, & -24 \\ -15, & 30 \end{bmatrix}.$$

Now let us generalize the algorithm for k-dimensional data. Let us assume that the dimensions of the convolution system $N_1, N_2, \ldots, N_k$ are powers of 2. Then the algorithm of inversion of the polynomial $H(z_1, z_2, \ldots, z_k)$ and its simultaneous multiplication by the polynomial $Y(z_1, z_2, \ldots, z_k)$ can be expressed as successive reductions of the response function (autoconvolutions) and convolutions of $Y(z_1, z_2, \ldots, z_k)$ with reduced response (crossconvolutions).

The $(p+1)-th$ reduction of the response in the direction k (autoconvolution) is

$$^{p+1}h_k(l_1,\ldots,l_{k-1},2^{p+1}\cdot l_k) = \sum_{i_1=0}^{N_1-1}\cdots\sum_{i_{k-1}=0}^{N_{k-1}-1}\sum_{i_k=0}^{\left(\frac{N_k}{2^p}\right)-1} {}^{p}h_k(i_1,\ldots,i_{k-1},2^p\cdot i_k,)\,{}^{p}h_k(m_1,\ldots,m_{k-1},m_k)\cdot(-1)^{i_k}, \quad (29)$$

where

$$m_1 = (l_1 - i_1) \bmod N_1, \cdots, m_{k-1} = (l_{k-1} - i_{k-1}) \bmod N_{k-1}, m_k = (2^{p+1}\cdot l_k - 2^p \cdot i_k) \bmod N_k,$$

$$l_1 \in \langle 0, N_1 - 1\rangle, \ldots, l_{k-1} \in \langle 0, N_{k-1} - 1\rangle, \quad l_k \in \left\langle 0, \left(\frac{N_k}{2^{p+1}}\right) - 1\right\rangle,$$

and the number of the reduction $p = 0,1,\ldots,\log_2 N_{k-1}$.

Next we derive the algorithm of crossconvolution. Let

$$^{0}n_k(i_1,i_2,\ldots,i_k) = y(i_1,i_2,\ldots,i_k),$$

where

$$i_1 \in \langle 0, N_1 - 1\rangle, \quad i_2 \in \langle 0, N_2 - 1\rangle, \ldots, i_k \in \langle 0, N_k - 1\rangle.$$

Then the calculation of the $(p+1)-th$ reduction of the signal n_k in the direction k is

$$^{p+1}n_k(l_1,\ldots,l_{k-1},l_k) = \sum_{i_1=0}^{N_1-1}\cdots\sum_{i_{k-1}=0}^{N_{k-1}-1}\sum_{i_k=0}^{\left(\frac{N_k}{2^p}\right)-1} {}^{p}n_k(i_1,\ldots,i_{k-1},2^p\cdot i_k,)\,{}^{p}h_k(m_1,\ldots,m_{k-1},m_k)\cdot(-1)^{i_k}, \quad (30)$$

where

$$m_1 = (l_1 - i_1) \bmod N_1, \cdots, m_{k-1} = (l_{k-1} - i_{k-1}) \bmod N_{k-1}, m_k = (l_k - 2^p \cdot i_k) \bmod N_k,$$

$$l_1 \in \langle 0, N_1 - 1\rangle, \ldots, l_{k-1} \in \langle 0, N_{k-1} - 1\rangle, \quad l_k \in \langle 0, N_k - 1\rangle,$$

and the number of the reduction $p = 0,1,\ldots,\log_2 N_{k-1}$.

We repeat the algorithm (29), (30) until the number of reductions is $\log_2 N_{k-1}$. Then we continue with the reductions of the response and the signal n_k in the direction $k-1$ (again using formulas (29), (30)). We proceed in this way until all the reductions of the response in all the directions are carried out, i.e., until the k-dimensional discrete signal of the response is reduced to the only non-zero element. It represents the value of the determinant of the system matrix h_k, hence

$$D = {}^{r}h_k(0,0,\ldots,0),$$

where

$$r = \sum_{i=1}^{k} \log_2 N_i .$$

The final solution is $\mathbf{x}_k = \frac{\mathbf{n}_k}{D}$.

3. Special linear systems

3.1 Error-free algorithm to solve integer Toeplitz system [8]

In this section we propose an algorithm to solve integer nonsymmetrical Toeplitz system, which is based on Levinson algorithm [9]. It removes rounding off errors. Their accumulation can, when using classic algorithms, deteriorate or destroy the solution. Toeplitz system of linear equations is defined

$$\begin{bmatrix} a_0, & a_{-1}, & a_{-2},\ldots, & a_{-N+1} \\ a_1, & a_0, & a_{-1},\ldots, & a_{-N+2} \\ \vdots & & & \vdots \\ a_{N-1}, & a_{N-2}, & a_{N-3},\ldots, & a_0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \tag{32}$$

or

$$\sum_{j=1}^{N} a_{i-j} x_j = y_i, \quad i = 1,2,\ldots,N . \tag{33}$$

For integer regular Toeplitz system one can write

$$\sum_{j=1}^{N} a_{i-j} \frac{x_j}{D_N} = y_i, \quad i = 1,2,\ldots,N. \tag{34}$$

In [8] we derived the algorithm for fast solution of integer Toeplitz system (34) using recursive procedure solving in each iteration step the system

$$\sum_{j=1}^{M} a_{i-j} \frac{v_j^{(M)}}{D_M} = y_i, \quad i = 1,2,\ldots,M; \quad M = 1,2,\ldots,N. \tag{35}$$

To determine quantities with the index $M+1$ we get

$$\begin{aligned} D_{M+1} &= D_M \cdot a_0 - \sum_{j=1}^{M} a_j b_j^{(M)}, \\ v_{M+1}^{(M+1)} &= \frac{y_{M+1} D_M - \sum_{j=1}^{M} a_{M+1-j} v_j^{(M)}}{\frac{a_0 D_M - \sum_{j=1}^{M} a_{M+1-j} b_{M+1-j}^{(M)}}{D_{M+1}}}, \\ c_{M+1}^{(M+1)} &= a_{M+1} D_M - \sum_{j=1}^{M} a_{M+1-j} c_j^{(M)}, \\ b_{M+1}^{(M+1)} &= a_{-M-1} D_M - \sum_{j=1}^{M} a_{j-M-1} b_j^{(M)}. \end{aligned} \tag{36}$$

To determine remaining components of the vectors **v, b, c** we have

$$v_j^{(M+1)} = \frac{1}{D_M}\left[v_j^{(M)} D_{M+1} - v_{M+1}^{(M+1)} b_{M+1-j}^{(M)}\right],$$
$$c_j^{(M+1)} = \frac{1}{D_M}\left[c_j^{(M)} D_{M+1} - c_{M+1}^{(M+1)} b_{M+1-j}^{(M)}\right], \tag{37}$$
$$b_j^{(M+1)} = \frac{1}{D_M}\left[b_j^{(M)} D_{M+1} - b_{M+1}^{(M+1)} c_{M+1-j}^{(M)}\right], \quad j = 1,2,\ldots,M,$$

with initial values

$$D_1 = a_0, v_1^{(1)} = y_1, c_1^{(1)} = a_1, b_1^{(1)} = a_{-1}. \tag{38}$$

From the above given formulas, one can see that all quantities are integers. After some iteration steps, however, their magnitudes increase rapidly, which complicates the realization of the calculation. To overcome this problem we can carry out the calculation in modulo arithmetic in different prime modulo classes $m_i, i = 1,2,\cdots,r$, so that it holds [10]

$$P = m_1 m_2 \ldots m_r, \tag{39}$$

where P must satisfy the condition

$$P > 2 \cdot max\left\{N^{N/2} \cdot M(A)^N, N(N-1)^{(N-1)/2} \cdot M(A)^{N-1} \cdot M(y)\right\},$$
$$M(A) = max|a_i|, \quad i \in \langle -N+1, N-1\rangle, M(y) = max|y_j|, \quad j \in \langle 1, N\rangle. \tag{40}$$

The calculation in the given modulo class is independent of other modulo classes and thus this model of implementation is well suited for parallel computing. By inverse conversion from residual representation, employing Chinese theorem [2], [9], one can calculate resulting vector **x** and determinant D.

Illustrative example:

Let us find the exact solution of the Toeplitz system

$$\begin{bmatrix} 1, & -1, & 2 \\ 3, & 1, & -1 \\ 2, & 3, & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \\ 1 \end{bmatrix}.$$

First we initialize the vectors and variables

$$\mathrm{a_p} = \left[a_1, a_2, 0\right]^T = \left[3,2,0\right]^T, \mathrm{a_n} = \left[a_{-1}, a_{-2}, 0\right]^T = \left[-1,2,0\right]^T, \mathrm{y} = \left[-1,3,1\right]^T$$
$$\mathrm{b} = \left[-1,-,-\right]^T, \mathrm{c} = \left[3,-,-\right]^T, \mathrm{v} = \left[-1,-,-\right]^T, D = 1.$$

With respect to (39) and (40) we choose the moduli $m_1 = 5, \quad m_2 = 7, \quad m_3 = 11.$ Carrying out the first iteration step yields

$$\mathrm{b}(mod\,5) = \left[3, \quad 1, \quad -\right]^T; \quad \mathrm{b}(mod\,7) = \left[0, \quad 1, \quad -\right]^T; \quad \mathrm{b}(mod\,11) = \left[4, \quad 1, \quad -\right]^T;$$
$$\mathrm{c}(mod\,5) = \left[0, \quad 3, \quad -\right]^T; \quad \mathrm{c}(mod\,7) = \left[5, \quad 0, \quad -\right]^T; \quad \mathrm{c}(mod\,11) = \left[5, \quad 4, \quad -\right]^T;$$
$$\mathrm{v}(mod\,5) = \left[2, \quad 1, \quad -\right]^T; \quad \mathrm{v}(mod\,7) = \left[2, \quad 6, \quad -\right]^T; \quad \mathrm{v}(mod\,11) = \left[2, \quad 6, \quad -\right]^T;$$
$$D(mod\,5) = 4; \qquad D(mod\,7) = 4; \qquad D(mod\,11) = 4.$$

After the second iteration step $(M = N-1 = 2)$ the calculation in this example is finished. The resulting values of vectors and determinant are then as follows

$$\mathrm{b}(mod\,5) = [3,\ 1,\ 0]^T;\quad \mathrm{b}(mod\,7) = [0,\ 1,\ 1]^T;\quad \mathrm{b}(mod\,11) = [4,\ 1,\ 4]^T;$$
$$\mathrm{c}(mod\,5) = [0,\ 3,\ 4]^T;\quad \mathrm{c}(mod\,7) = [5,\ 0,\ 4]^T;\quad \mathrm{c}(mod\,11) = [5,\ 4,\ 0]^T;$$
$$\mathrm{v}(mod\,5) = [1,\ 3,\ 2]^T;\quad \mathrm{v}(mod\,7) = [2,\ 3,\ 3]^T;\quad \mathrm{v}(mod\,11) = [5,\ 3,\ 4]^T;$$
$$D(mod\,5) = 3;\qquad D(mod\,7) = 2;\qquad D(mod\,11) = 1.$$

By conversion of the vector $\mathbf{v}$ and the determinant D from residue class code one obtains

$$\mathbf{v} = [16,\ 3,\ 367]^T,\quad D = 23.$$

Positive numbers are represented in the range $\langle 0,(P-1)/2\rangle$ and negative ones in the range $\langle (P-1)/2+1, P-1\rangle$. Then the negative values are calculated by subtracting P. Then the final solution is

$$\mathbf{x} = \frac{1}{23}[16,\ 3,\ -18]^T.$$

3.2 An algorithm to solve Hilbert system of linear equations exactly [11]

Typical example of extremely ill-conditioned matrices are Hilbert matrices. In literature one can find the examples of Hilbert matrices with the elements

$$a_{i,j} = \frac{1}{i+j-1};\quad i,j \in \langle 1,N\rangle.$$

We shall consider more general Hilbert systems with elements

$$a_{i,j} = \frac{1}{d_{i+j-1}};\quad i,j \in \langle 1,N\rangle.$$

Then for $N = 3$ we can write

$$\begin{bmatrix} \frac{1}{b_1}, & \frac{1}{b_2}, & \frac{1}{b_3} \\ \frac{1}{b_2}, & \frac{1}{b_3}, & \frac{1}{b_4} \\ \frac{1}{b_3}, & \frac{1}{b_4}, & \frac{1}{b_5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

or after exchanging the elements of the vector $\mathbf{x}$

$$\begin{bmatrix} \frac{1}{b_3}, & \frac{1}{b_2}, & \frac{1}{b_1} \\ \frac{1}{b_4}, & \frac{1}{b_3}, & \frac{1}{b_2} \\ \frac{1}{b_5}, & \frac{1}{b_4}, & \frac{1}{b_3} \end{bmatrix} \begin{bmatrix} x_3 \\ x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} \frac{1}{s_0}, & \frac{1}{s_{-1}}, & \frac{1}{s_{-2}} \\ \frac{1}{s_1}, & \frac{1}{s_0}, & \frac{1}{s_{-1}} \\ \frac{1}{s_2}, & \frac{1}{s_1}, & \frac{1}{s_0} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} R_0, & R_{-1}, & R_{-2} \\ R_1, & R_0, & R_{-1} \\ R_2, & R_1, & R_0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}.$$

However this represents Toeplitz system of linear equations. Assuming that all quantities are integers the determinant of the matrix can be written in the form of rational fraction. For $N=3$ one obtains

$$D=\frac{1}{s_0^3}+\frac{1}{s_2 s_{-1}^2}+\frac{1}{s_1^2 s_{-2}}-\frac{1}{s_2 s_0 s_{-2}}-\frac{2}{s_1 s_0 s_{-1}}=\frac{{}^{C}D}{s_{-2}^1 s_{-1}^2 s_0^3 s_1^2 s_2^1}=\frac{{}^{C}D}{{}^{D}D},$$

where left upper index C denotes numerator and D denominator. Without being interested in the numerator for the denominator of the determinant in general case one can write

$${}^{D}D=s_{-N+1}^1 s_{-N+2}^2 \ldots s_{-1}^{N-1} s_0^N s_1^{N-1} \cdots s_{N-2}^2 s_{N-1}^1 .$$

We introduce pairs of expressions of all quantities for both numerator and denominator, respectively. First let us determine quantities with the index $M+1$

$${}^{D}b_{M+1}^{(M+1)}=\prod_{i=-M-1}^{M-1} s_i^{M+1-|i+1|}; \qquad {}^{C}b_{M+1}^{(M+1)}={}^{D}b_{M+1}^{(M+1)}\left[\frac{{}^{C}D_M}{s_{-M-1}\,{}^{D}D_M}-\sum_{j=1}^{M}\frac{{}^{C}b_j^{(M)}}{s_{j-M-1}\,{}^{D}b_j^{(M)}}\right];$$

$${}^{D}c_{M+1}^{(M+1)}=\prod_{i=-M+1}^{M+1} s_i^{M+1-|i-1|}; \qquad {}^{C}c_{M+1}^{(M+1)}={}^{D}c_{M+1}^{(M+1)}\left[\frac{{}^{C}D_M}{s_{M+1}\,{}^{D}D_M}-\sum_{j=1}^{M}\frac{{}^{C}c_j^{(M)}}{s_{M+1-j}\,{}^{D}c_j^{(M)}}\right];$$

$${}^{D}v_{M+1}^{(M+1)}={}^{D}c_{M+1}^{(M+1)}\frac{\prod_{i=1}^{M+1}{}^{D}y_i}{\prod_{i=1}^{M+1}s_i}; \quad {}^{C}v_{M+1}^{(M+1)}={}^{D}v_{M+1}^{(M+1)}\left[\frac{{}^{C}y_{M+1}\,{}^{C}D_M}{{}^{D}y_{M+1}\,{}^{D}D_M}-\sum_{j=1}^{M}\frac{{}^{C}v_j^{(M)}}{s_{M+1-j}\,{}^{D}v_j^{(M)}}\right];$$

$${}^{D}D_{M+1}=\prod_{i=-M}^{M} s_i^{M+1-|i|}; \qquad {}^{C}D_{M+1}={}^{D}D_{M+1}\left[\frac{{}^{C}D_M}{s_0\,{}^{D}D_M}-\sum_{j=1}^{M}\frac{{}^{C}b_j^{(M)}}{s_{M+1-j}\,{}^{D}b_j^{(M)}}\right].$$

Then we determine remaining components of the vectors **b, c, v**

$${}^{D}b_j^{(M+1)}={}^{D}D_{M+1}\frac{\prod_{i=0}^{j-1}s_{-M-1+i}}{\prod_{i=0}^{j-1}s_i}; \quad {}^{C}b_j^{(M+1)}=\frac{{}^{D}b_j^{(M+1)}\,{}^{D}D_M}{{}^{C}D_M}\left[\frac{{}^{C}D_{M+1}\,{}^{C}b_j^{(M)}}{{}^{D}D_{M+1}\,{}^{D}b_j^{(M)}}-\frac{{}^{C}b_{M+1}^{(M+1)}\,{}^{C}c_{M+1-j}^{(M)}}{{}^{D}b_{M+1}^{(M+1)}\,{}^{D}c_{M+1-j}^{(M)}}\right];$$

$${}^{D}c_j^{(M+1)}={}^{D}D_{M+1}\frac{\prod_{i=0}^{j-1}s_{M+1-i}}{\prod_{i=0}^{j-1}s_{-i}}; \quad {}^{C}c_j^{(M+1)}=\frac{{}^{D}c_j^{(M+1)}\,{}^{D}D_M}{{}^{C}D_M}\left[\frac{{}^{C}D_{M+1}\,{}^{C}c_j^{(M)}}{{}^{D}D_{M+1}\,{}^{D}c_j^{(M)}}-\frac{{}^{C}c_{M+1}^{(M+1)}\,{}^{C}b_{M+1-j}^{(M)}}{{}^{D}c_{M+1}^{(M+1)}\,{}^{D}b_{M+1-j}^{(M)}}\right];$$

$${}^{D}v_j^{(M+1)}={}^{D}c_j^{(M+1)}\frac{{}^{D}v_{M+1}^{(M+1)}}{{}^{D}c_{M+1}^{(M+1)}}; \quad {}^{C}b_j^{(M+1)}=\frac{{}^{D}v_j^{(M+1)}\,{}^{D}D_M}{{}^{C}D_M}\left[\frac{{}^{C}D_{M+1}\,{}^{C}v_j^{(M)}}{{}^{D}D_{M+1}\,{}^{D}v_j^{(M)}}-\frac{{}^{C}v_{M+1}^{(M+1)}\,{}^{C}b_{M+1-j}^{(M)}}{{}^{D}v_{M+1}^{(M+1)}\,{}^{D}b_{M+1-j}^{(M)}}\right].$$

The initial values were set

$${}^{D}D_1=s_0;\ {}^{C}D=1;\ {}^{D}v_1^{(1)}={}^{D}y_1;\ {}^{C}v_1^{(1)}={}^{C}y_1;\ {}^{D}b_1^{(1)}=s_{-1};\ {}^{C}b_1^{(1)}=1;\ {}^{D}c_1^{(1)}=s_1;\ {}^{C}c_1^{(1)}=1.$$

3.3 Error-free algorithm to solve Vandermonde system of linear equations [12]

Let us have Vandermonde system

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ a_1 & a_2 & a_3 & \cdots & a_N \\ a_1^2 & a_2^2 & a_3^2 & \cdots & a_N^2 \\ \vdots & \vdots & \vdots & & \vdots \\ a_1^{N-1} & a_2^{N-1} & a_3^{N-1} & \cdots & a_N^{N-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} = A \cdot \mathbf{x}. \tag{41}$$

It is well known that the determinant of the system matrix is

$$D = \prod_{j=1}^{N-1} \prod_{i=j+1}^{N} (a_i - a_j) \tag{42}$$

and the elements of inverse matrix can be expressed using polynomial coefficients [12]

$$P_j(x) = \prod_{i=1, i\neq j}^{N} \frac{x - a_i}{a_j - a_i} = \sum_{k=1}^{N} B_{j,k} x^{k-1}. \tag{43}$$

Analogously to the algorithm given in [13] we determine coefficients of the polynomial

$$P(x) = \prod_{i=1}^{N} (x - a_i) (mod\ M) = x^N + c_N x^{N-1} + \cdots + c_2 x + c_1 (mod\ M). \tag{44}$$

Polynomials from (43) modulo M can be expressed

$$P_j(x) = \frac{P(x)}{(x - a_j)} \cdot \frac{1}{\prod_{i=1, i\neq j}^{N} (a_j - a_i)} (mod\ M) = \frac{\sum_{k=1}^{N} z_{j,k} x^{k-1}}{\prod_{i=1, i\neq j}^{N} (a_j - a_i)} (mod\ M). \tag{45}$$

Then one can derive

$$\begin{aligned} & z_{j,N} = 1, \\ & z_{j,N-1} = (c_N + z_{j,N} \cdot a_j) (mod\ M), \\ & \vdots \\ & z_{j,N-i} = (c_{N-i+1} + z_{j,N-i+1} \cdot a_j) (mod\ M) \\ & i \in \langle 1, N-1 \rangle, \quad j \in \langle 1, N \rangle. \end{aligned} \tag{46}$$

Resulting inverse matrix modulo M is

$$A^{-1} (mod\ M) = \begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & d_N \end{bmatrix}^{-1} \begin{bmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,N} \\ z_{2,1} & z_{2,2} & \cdots & z_{2,N} \\ \vdots & \vdots & & \vdots \\ z_{N,1} & z_{N,2} & \cdots & z_{N,N} \end{bmatrix} (mod\ M) = B_M, \tag{47}$$

where

$$d_j = \prod_{i=1, i\neq j}^{N} (a_j - a_i) (mod\ M).$$

We can employ iterative scheme of error-free solution of linear equation system from the section 2.1.

Illustrative example:

Let us have the Vandermonde system

$$\begin{bmatrix} 1,1, & 1, & 1 \\ 1,3, & 5, & 4 \\ 1,9, & 25, & 16 \\ 1,27, & 125, & 64 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 0 \end{bmatrix}.$$

The determinant of the matrix, according to (42), is -48. We shall use modulus $M = 5$. Then $D_M = 2$. Using the relations (45), (46), (47) yields inverse matrix in modulo class 5

$$B_M = \begin{bmatrix} -24,0, & 0, & 0 \\ 0, & 4, & 0, & 0 \\ 0, & 0, & 8, & 0 \\ 0, & 0, & 0, & -3 \end{bmatrix} \begin{bmatrix} -60,47, & -12,1 \\ -20,29, & -10,1 \\ -12,19, & -8, & 1 \\ -15,23, & -9, & 1 \end{bmatrix} (mod\ 5) = \begin{bmatrix} 0,2,3,1 \\ 0,1,0,4 \\ 1,3,4,2 \\ 0,4,3,3 \end{bmatrix}.$$

Then the procedure to calculate the exact solution is as follows

$$\mathbf{ym_0} = \mathbf{y} = [0,1,2,0]^{\mathbf{T}}; \quad \mathbf{x_M} = B_M \mathbf{ym_0} (mod\ 5) = \begin{bmatrix} 0,2,3,1 \\ 0,1,0,4 \\ 1,3,4,2 \\ 0,4,3,3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 2 \\ 0 \end{bmatrix} (mod\ 5) = \begin{bmatrix} 3 \\ 1 \\ 1 \\ 0 \end{bmatrix};$$

$$\mathbf{x_0} = D_M \mathbf{x_M} = 2[3,1,1,0]^{\mathbf{T}} (mod\ 5) = [1,2,2,0]^{\mathbf{T}}.$$

Next we calculate vectors

$$\mathbf{y}_0' = A\mathbf{x_0} = \begin{bmatrix} 1,1, & 1, & 1 \\ 1,3, & 5, & 4 \\ 1,9, & 25, & 16 \\ 1,27, & 125, & 64 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 17 \\ 69 \\ 305 \end{bmatrix}; \mathbf{y_1} = \frac{1}{5} \left(\begin{bmatrix} 0 \\ -48 \\ -96 \\ 0 \end{bmatrix} - \begin{bmatrix} 5 \\ 17 \\ 69 \\ 305 \end{bmatrix} \right) = \begin{bmatrix} -1 \\ -13 \\ -33 \\ -61 \end{bmatrix};$$

$$\mathbf{ym_1} = [4,2,2,4]^T$$

1-st iteration step

$$\mathbf{x_1} = B_M \mathbf{ym_1} (mod\ 5) = \begin{bmatrix} 0,2,3,1 \\ 0,1,0,4 \\ 1,3,4,2 \\ 0,4,3,3 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ 2 \\ 4 \end{bmatrix} (mod\ 5) = \begin{bmatrix} 4 \\ 3 \\ 1 \\ 1 \end{bmatrix}.$$

Again let us suppose that the positive numbers are represented in the range $\langle 0,(M-1)/2 \rangle$ and the negative ones in the range $\langle (M-1)/2+1, M-1 \rangle$. Then we obtain the negative values for numbers greater than $(M-1)/2$ by subtracting the modulus

$$\mathbf{x_1} = [-1,-2,1,1]^{\mathrm{T}}.$$

Further we calculate vectors

$$\mathbf{y}_1' = A\mathbf{x_1} = [-1,2,22,134]^T; \quad \mathbf{y}_2 = 5^{-1}[0,3,-11,-39]^T; \quad \mathbf{ym}_2 = [0,2,4,1]^T.$$

2-nd iteration step

We continue in calculations with

$\mathbf{x_2} = B_M \mathbf{ym_2} (mod\ 5) = [2,1,3,4]^T$.

After adjustment of elements greater than $(M-1)/2$ we have

$\mathbf{x_2} = [2,1,-1,-2]^T$.

Again we calculate vectors

$\mathbf{y}_2' = A\mathbf{x_2} = [0,-8,-46,-224]^T$; $\mathbf{y}_3 = [0,1,7,37]^T$; $\mathbf{ym}_3 = [0,1,2,2]^T$.

3-rd iteration step

Finally we calculate

$\mathbf{x_3} = B_M \mathbf{ym_3} (mod\ 5) = [0,-1,0,1]^T$.

We calculate vectors

$\mathbf{y}_3' = A\mathbf{x_3} = [0,1,7,37]^T$; $\mathbf{y}_4 = [0,0,0,0]^T$.

The zero vector $\mathbf{y_4}$ indicates the end of iterations. The result is

$$\mathbf{x} = \frac{1}{-48}\left\{\begin{bmatrix}1\\2\\2\\0\end{bmatrix} + 5\cdot\begin{bmatrix}-1\\-2\\1\\1\end{bmatrix} + 5^2\cdot\begin{bmatrix}2\\1\\-1\\-2\end{bmatrix} + 5^3\cdot\begin{bmatrix}0\\-1\\0\\1\end{bmatrix}\right\} = \frac{1}{-48}\begin{bmatrix}46\\-108\\-18\\80\end{bmatrix}.$$

4. General linear systems

4.1 Parallel error-free algorithm [14, 15]

Let us have system of linear equations, where all the elements of the matrix A and vector $\mathbf{y}$ are integers

$$y(n) = \sum_{m=1}^{N} a(n,m)x(m), \quad n = 1,2,\cdots,N$$

or

$$\begin{bmatrix} a_{1,1}, & a_{1,2}, & a_{1,3},\ldots, & a_{1,N} \\ a_{2,1}, & a_{2,2}, & a_{2,3},\ldots, & a_{2,N} \\ \vdots & & & \vdots \\ a_{N,1}, & a_{N,2}, & a_{N,3},\ldots, & a_{N,N} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = A\mathbf{x}=\mathbf{y}.$$

Let us split the calculation into several residue classes satisfying conditions (39), (40). Then in each residue class we can carry out separately the calculations

$A\mathbf{x}(mod\ m_i) = \mathbf{y}, \quad i \in \langle 1,r \rangle.$

Calculations in residue classes can be carried out in parallel. Gauss-Jordan elimination in residue class m can be described by the following procedure. Let

$a_{i,j}^{(0)} = \left[a_{i,j} \right] (mod\ m); \quad i \in \{1,2,\ldots,n\}, \quad j \in \{1,2,\ldots,n+1\}.$

For $k = 1,2,\ldots,n$ (k denotes elimination step) do:

1. search for such $i \neq c_f \quad (f < k)$ that $a_{i,1} \neq 0, \quad c_k = i,$
2. $$a_{i,j}^{(k)} = \left[a_{i,j+1}^{(k-1)} \cdot \left(a_{i,1}^{(k-1)} \right)^{-1} \right] (mod\ m), \quad j \in \{1,2,\ldots,n+1-k\}, \tag{49}$$
3. for $l = 1,2,\ldots,n$ and $l \neq i$ do:
$$a_{l,j}^{(k)} = \left[a_{l,j+1}^{(k-1)} - a_{l,1}^{(k-1)} \cdot a_{i,1}^{(k)} \right] (mod\ m), \quad j \in \{1,2,\ldots,n+1-k\}. \tag{50}$$

The determinant in the residue class m is obtained as the modulo product of the elements $a_{i,1}^{(k-1)}$ (see (49))

$$D(mod\ m) = D_m = \left[(-1)^J \left[\prod_{k=1}^{n} a_{k,1}^{(k-1)} \right] \right] (mod\ m), \tag{51}$$

where J is the total number of exchanges of the elements. Then, the sought solution of (48) in the residue class m is given by

$$\mathbf{x}(mod\ m) = \mathbf{x_m} = \left[\frac{D}{D} \mathbf{x_m} \right] (mod\ m) = \frac{1}{D} \left[D_m \cdot \mathbf{x_m} \right] (mod\ m). \tag{52}$$

After the calculation of the solution of a linear equations set in the modular arithmetic is finished we have $n+1$ element vectors

$$\left(D(mod\ m_k), x_1(mod\ m_k), \ldots, x_n(mod\ m_k) \right), \quad k = 1,2,\ldots,r. \tag{53}$$

The resulting integer values $D, x_1, x_2, \ldots, x_n$ can be obtained by the inverse conversion of the vectors (53) from the residue class code. Since we use the known algorithm (given e.g. in [16]) we describe it very briefly from algorithm point of view. We suppose we have a vector of moduli

$$\beta_0 = \left(m_1, m_2, \ldots, m_r \right) \tag{54}$$

and a vector of the residue representation of the integer x_j

$$t_{j,1} = \left(x_j(mod\ m_1), x_j(mod\ m_2), \ldots, x_j(mod\ m_r) \right). \tag{55}$$

Further let us denote

$$\beta_k = \left(m_{k+1}, m_{k+2}, \ldots, m_r \right)$$
$$t_{j,k} = \left(t_{j,k}(k), t_{j,k}(k+1), \ldots, t_{j,k}(r) \right)$$
$$d_{j,k} = \left(t_{j,k}(k)(mod\ m_k), \ldots, t_{j,k}(k)(mod\ m_r) \right).$$

If we define the vector $t_{j,k+1}$ to be

$$t_{j,k+1} = \left[\left(t_{j,k} - d_{j,k} \right) \cdot m_k^{-1} \right] (mod\ \beta_k),$$

then the resulting sought number is

$$x_j = t_{j,1}(1) + m_1 \left(t_{j,2}(2) + m_2 \left[t_{j,3}(3) + \cdots + m_{r-1} \cdot t_{j,r}(r) \right] \right). \tag{56}$$

4.2 Sequential algorithm to solve general system of linear equations [17]

So far we have mentioned two basic algorithm classes to solve linear equations exactly

- iterative algorithm (applied in section 2.1 – Fermat transform, section 3.3 – Vandermonde system)
- parallel algorithm (applied in section 3.1 – Toeplitz system, section 4.1 – general system of linear equations)

In this section we want to demonstrate that sequential iterative algorithm can be applied also for general systems of linear integer equations. Let us define the following algorithm:

Algorithm A

a. Let $\mathbf{y_{m0}} = \mathbf{y}$.
b. Calculate vector $\mathbf{x}_M = A^{-1}\mathbf{y_{m0}} \left(mod\ M\right)$.
c. It holds $\mathbf{x_0} = D_M \mathbf{x}_M \left(mod\ M\right)$.
d. Let $\mathbf{x} = \mathbf{x}_0$.
e. Calculate vectors

$$\mathbf{y_0}^{'} = A\mathbf{x}_0, \quad \mathbf{y}_1 = \frac{1}{M}\left(\mathbf{y_{m0}} \cdot D - \mathbf{y_0}^{'}\right), \quad \mathbf{y_{m1}} = \mathbf{y}_1 \left(mod\ M\right). \tag{57}$$

f. $j = 1$.
g. Calculate $\mathbf{x}_j = A^{-1}\mathbf{y_{m1}} \left(mod\ M\right)$. (58)
h. For $i = 0,1,\ldots,N-1$ (N is the size of vectors $\mathbf{x, y}$) calculate

$$x(i) = x(i) + x_j(i) M^j.$$

i. Calculate vectors

$$\mathbf{y_j}^{'} = A\mathbf{x_j}, \quad \mathbf{y_{j+1}} = \frac{\mathbf{y_j} - \mathbf{y_j}^{'}}{M}, \quad \mathbf{y_{m,j+1}} = \mathbf{y_{j+1}} \left(mod\ M\right).$$

j. If for all $i = 0,1,\ldots,N-1$, $y_{j+1}(i) = 0$, finish the calculation. If not, increment j and repeat the algorithm from point g. on.

The resulting solution of system (48) is simply calculated as $\mathbf{x} = \frac{\mathbf{x}}{D}$. Then determinant of the system of linear equations can be calculated using the formula

$$D = \prod_{j=1}^{N-1}\left[a(j,j) - \sum_{i=0}^{j-1} a(j,i) \cdot k_j(i)\right], \tag{59}$$

where the coefficient vectors $\mathbf{k_j}$ are calculated by exact solution of sets

$$A_{j,j} \cdot \mathbf{k_j} = A_{j,1} \quad j = 1,2,\ldots,N-1, \tag{60}$$

using the Algorithm A.

Next problem in the algorithm A is calculation of the inverse matrix in modulo arithmetic. Let us define iterative procedure where

$$A_{j+1,j+1} = \begin{bmatrix} A_{j,j} & A_{j,1} \\ A_{1,j} & A_{1,1} \end{bmatrix}, \qquad A^{-1}_{j+1,j+1} \left(mod\ M\right) = \begin{bmatrix} B_{j,j} & B_{j,1} \\ B_{1,j} & B_{1,1} \end{bmatrix}. \tag{61}$$

Then one can derive

$$
\begin{aligned}
B_{1,1} &= \left[A_{1,1} - A_{1,j} \cdot A_{j,j}^{-1} \cdot A_{j,1} \right]^{-1} (mod\ M), \\
B_{j,1} &= -A_{j,j}^{-1} \cdot A_{j,1} \cdot B_{1,1} (mod\ M), \\
B_{1,j} &= -B_{1,1} \cdot A_{1,j} \cdot A_{j,j}^{-1} (mod\ M), \\
B_{j,j} &= A_{j,j}^{-1} + B_{j,1} \cdot B_{1,1}^{-1} \cdot B_{1,j} (mod\ M).
\end{aligned} \tag{62}
$$

4.3 Sequential error-free algorithm to solve system of polynomial equations [18]

Sequential iterative algorithm can be successively applied also for polynomial systems of linear integer equations

$$
\begin{aligned}
&A(s) \cdot \mathbf{x}(s) = \mathbf{y}(s), \\
&a_{i,j}(s) = {}^{p-1}a_{i,j} \cdot s^{p-1} + \cdots + {}^{1}a_{i,j} \cdot s^{1} + {}^{0}a_{i,j}, \quad y_i(s) = {}^{p-1}y_i \cdot s^{p-1} + \cdots + {}^{1}y_i \cdot s^{1} + {}^{0}y_i.
\end{aligned} \tag{63}
$$

Assuming the coefficients being integers, the solution of the system given by (63) can be expressed as

$$
\mathbf{x}(s) = \frac{1}{D(s)} \cdot \left[\left(s^p - 1\right) \cdot \mathbf{x}_0(s) + \left(s^p - 1\right)^1 \cdot \mathbf{x}_1(s) + \cdots + \left(s^p - 1\right)^k \cdot \mathbf{x}_k(s) \right], \tag{64}
$$

and vectors

$$
\mathbf{x}_\mathbf{i}(s) = \left[M^0 \cdot \mathbf{x}_{i0}(s) + M^1 \cdot \mathbf{x}_{i1}(s) + \cdots + M^m \cdot \mathbf{x}_{im}(s) \right], \tag{65}
$$

where M is a prime modulus D is determinant of the matrix A and m, k are finite integers. Let us suppose for the moment that the determinant $D(s)$ (polynomial) is known. We denote polynomial modulus as

$$
P(s) = s^p - 1.
$$

The algorithms for the determinant and inverse matrix of polynomial system can be derived analogously to relations (59-62). For details we refer to [18]. New problem is the calculation of inverse polynomial. In what follows we outline briefly this algorithm.

We have to find polynomial $b(s)$ to polynomial $a(s)$ so that

$$
a(s) \cdot b(s) \cdot \left(mod\ P(s)\right)(mod\ M) = 1. \tag{66}
$$

According to [2], the cyclic convolution using polynomial algebra concept is

$$
c_l = \sum_{i=0}^{p-1} a_i \cdot b_{l-i}, \tag{67}
$$

where indices $l - i$ are calculated modulo p. From what is given above, it follows that solution of equation (66) represents the solution of linear system

$$
\begin{bmatrix} a(0), & a(p-1), & \cdots & a(1) \\ a(1), & a(0), & \cdots & a(2) \\ \vdots & \vdots & & \vdots \\ a(p-1), & a(p-2), & \cdots & a(0) \end{bmatrix} \cdot \begin{bmatrix} b(0) \\ b(1) \\ \vdots \\ b(p-1) \end{bmatrix} (mod\ M) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{68}
$$

in residual class M. Inverse polynomial $b(s)$ can be obtained by successive reductions of polynomial $a(s)$ or vector $\mathbf{a}$. For $r + 1 - st$ reduction it holds

$$ {}^{r+1}a\left(2^{r+1}\cdot l\right)=\sum_{i=0}^{\left(N/2^{r}\right)-1}{}^{r}a\left(2^{r}\cdot i\right)\cdot{}^{r}a\left(m\right)\cdot\left(-1\right)^{i}\,(mod\ M\ ), $$

where

$$ m=\left(2^{r+1}\cdot l-2^{r}\cdot i\right)(mod\ N\ );\quad l\in\left\langle 0,\left(\frac{N}{2^{r+1}}\right)-1\right\rangle, $$

and number of reduction $r=0,1,\ldots,log_2\ N$. Let ${}^{0}\mathbf{n}=\left[1,0,\cdots,0\right]^{T}$, and

$$ {}^{r+1}n\left(l\right)=\sum_{i=0}^{\left(N/2^{r}\right)-1}{}^{r}n\left(2^{r}\cdot i\right)\cdot{}^{r}a\left(m\right)\cdot\left(-1\right)^{i}\,(mod\ M\ ), $$

where

$$ m=\left(l-2^{r}\cdot i\right)(mod\ N\ );\quad l\in\left\langle 0,n-1\right\rangle, $$

and number of reduction $r=0,1,\ldots,\log_2 N-1.$ The resulting solution of polynomial inversion is

$$ \mathbf{b}=\frac{{}^{t}\mathbf{n}}{{}^{t}a\left(0\right)}\cdot(mod\ M\ ), $$

where $t=\log_2 N.$

Illustrative example:

Let us have linear system consisting of polynomials

$$ \begin{bmatrix} s+1, & 3s \\ 2s, & s^2+1 \end{bmatrix}\cdot\mathbf{x}\left(s\right)=\begin{bmatrix} s^2+1 \\ 2 \end{bmatrix}. $$

We shall use polynomial modulus $P\left(s\right)=s^3-1$ and numerical modulus $M=5$. Further we shall assume that we know the determinant $D\left(s\right)=s^3-5s^2+s+1,$ the determinant in the residue class $P(s)$ and the residue class M

$$ Dpm\left(s\right)=D\left(s\right)\left(mod\left(s^3-1\right)\right)\left(mod\ 5\right)=s+2\,, \tag{69} $$

and the inverse matrix in the residue class $P(s)$ and the residue class M

$$ B\left(s\right)=A^{-1}\left(s\right)\cdot\left(mod\left(s^3-1\right)\right)\left(mod\ 5\right)=\begin{bmatrix} s+3, & 4s^2+2s+3 \\ s^2+3s+2, & s^2+3s \end{bmatrix}. $$

We calculate

$$ \mathbf{x}_p\left(s\right)=\begin{bmatrix} s+3, & 4s^2+2s+3 \\ s^2+3s+2, & s^2+3s \end{bmatrix}\cdot\begin{bmatrix} s^2+1 \\ 2 \end{bmatrix}\cdot\left(mod\left(s^3-1\right)\right)\cdot\left(mod\ 5\right)=\begin{bmatrix} s^2 \\ 0 \end{bmatrix}. $$

Using $Dpm\left(s\right)$ from (69) we get

$$ \mathbf{x}_{00}\left(s\right)=Dpm\left(s\right)\cdot\begin{bmatrix} s^2 \\ 0 \end{bmatrix}\left(mod\left(s^3-1\right)\right)\left(mod\ 5\right)=\begin{bmatrix} 2s^2 \\ 0 \end{bmatrix}. $$

We initialize vectors and control variables

$$\mathbf{T}(s)=\mathbf{y}(s)\cdot D(s)=\begin{bmatrix}s^2+1\\2\end{bmatrix}\cdot\left(s^3-5s^2+s+1\right)=\begin{bmatrix}s^5-5s^4+2s^3-4s^2+s+1\\2s^3-10s^2+2s+2\end{bmatrix},$$

$$j=0,k=0,\mathbf{x}(s)=\begin{bmatrix}2s^2+1\\0\end{bmatrix}.$$

We calculate vectors

$$\mathbf{y}_{\mathbf{00}}^{'}(s)=A(s)\cdot\mathbf{x}_{00}(s)=\begin{bmatrix}s+1, & 3s\\2s, & s^2+1\end{bmatrix}\cdot\begin{bmatrix}2s^2+1\\0\end{bmatrix}=\begin{bmatrix}2s^3+2s^2+s+1\\4s^3+2s\end{bmatrix},$$

$$\mathbf{T}(s)=\mathbf{T}(s)-\mathbf{y}_{\mathbf{00}}^{'}(s)\cdot M^0=\begin{bmatrix}s^5-5s^4-6s^2\\-2s^3-10s^2+2\end{bmatrix},$$

$$\mathbf{y}_{\mathbf{01}}(s)=\frac{\mathbf{T}(s)\left(mod\left(s^3-1\right)\right)}{5}=\frac{1}{5}\cdot\begin{bmatrix}-5s^2-5s\\-10s^2\end{bmatrix}=\begin{bmatrix}-s^2-s\\-2s^2\end{bmatrix},$$

$$\mathbf{y}_{\mathbf{p,01}}(s)=\mathbf{y}_{\mathbf{01}}(s)(mod\ 5)=\begin{bmatrix}4s^2+4s\\3s^2\end{bmatrix}.$$

The vector $\mathbf{y}_{\mathbf{01}}(s)\neq 0$, thus we set $k=1$

$$\mathbf{x}_{\mathbf{01}}(s)=B(s)\cdot\mathbf{y}_{\mathbf{p,01}}(s)\left(mod\left(s^3-1\right)\right)(mod\ 5)=$$

$$=\begin{bmatrix}s+3, & 4s^2+2s+3\\s^2+3s+2, & s^2+3s\end{bmatrix}\cdot\begin{bmatrix}4s^2+4s\\3s^2\end{bmatrix}\left(mod\left(s^3-1\right)\right)(mod\ 5)==\begin{bmatrix}4s\\0\end{bmatrix}(mod\ 5)=\begin{bmatrix}-s\\0\end{bmatrix}.$$

If any coefficient of the solution is greater than $(M-1)/2$, it is considered negative and according to the rules of modulo arithmetic its negative value is calculated by subtracting modulus M . We update intermediate result

$$\mathbf{x}(s)=\begin{bmatrix}2s^2+1\\0\end{bmatrix}+\begin{bmatrix}-s\\0\end{bmatrix}\cdot 5^1=\begin{bmatrix}2s^2-5s+1\\0\end{bmatrix}.$$

Further, we calculate appropriate vectors

$$\mathbf{y}_{\mathbf{01}}^{'}(s)=a(s)\cdot\mathbf{x}_{\mathbf{01}}(s)=\begin{bmatrix}s+1, & 3s\\2s, & s^2+1\end{bmatrix}\cdot\begin{bmatrix}-s\\0\end{bmatrix}=\begin{bmatrix}-s^2-s\\-2s^2\end{bmatrix},$$

$$\mathbf{T}(s)=\mathbf{T}(s)-5^1\cdot\mathbf{y}_{\mathbf{01}}^{'}(s)=\begin{bmatrix}s^5-5s^4-s^2+5s\\-2s^3+2\end{bmatrix},$$

$$\mathbf{y}_{\mathbf{02}}(s)=\frac{\mathbf{T}(s)\left(mod\left(s^3-1\right)\right)}{5}=\mathbf{0},\quad \mathbf{y}_{\mathbf{p,02}}(s)=\mathbf{0}.$$

The vectors $\mathbf{y}_{\mathbf{p,02}}(s)=\mathbf{0},\mathbf{T}(s)\neq 0$. Therefore we set control variables $j=1,\ k=0$. Then we calculate

$$\mathbf{T}(s)=\frac{\mathbf{T}(s)}{s^3-1}=\begin{bmatrix}s^2-5s\\-2\end{bmatrix},\qquad \mathbf{y}_{\mathbf{p,10}}(s)=\mathbf{T}(s)(mod\ 5)=\begin{bmatrix}s^2\\3\end{bmatrix}.$$

Further we calculate next particular solution

$$\mathbf{x_{10}}(s)=\begin{bmatrix} s+3 & 4s^2+2s+3 \\ s^2+3s+2 & s^2+3s \end{bmatrix}\cdot\begin{bmatrix} s^2 \\ 3 \end{bmatrix}\cdot\left(mod\left(s^3-1\right)\cdot(mod\ 5)\right)=\begin{bmatrix} s \\ 3 \end{bmatrix}\cdot(mod\ 5)=\begin{bmatrix} s \\ -2 \end{bmatrix},$$

which we add to intermediate result

$$\mathbf{x}(s)=\begin{bmatrix} 2s^2-5s+1 \\ 0 \end{bmatrix}+\begin{bmatrix} s \\ -2 \end{bmatrix}\cdot\left(s^3-1\right)=\begin{bmatrix} s^4+2s^2-6s+1 \\ -2s^3+2 \end{bmatrix}.$$

Again we repeat and calculate

$$\mathbf{y_{10}'}(s)=A(s)\cdot\mathbf{x_{10}}(s)=\begin{bmatrix} s+1, & 3s \\ 2s, & s^2+1 \end{bmatrix}\cdot\begin{bmatrix} s \\ -2 \end{bmatrix}=\begin{bmatrix} s^2-5s \\ -2 \end{bmatrix},$$

$$\mathbf{T}(s)=\mathbf{T}(s)-\mathbf{y_{10}'}(s)\cdot 5^0=\mathbf{0},\ \mathbf{y_{11}}(s)=\mathbf{0},\ \mathbf{y_{p,11}}(s)=\mathbf{0}.$$

Both vectors $\mathbf{y_{11}}(s)$ and $\mathbf{T}(s)$ equal the zero vectors. It means that we finish the calculation. The resulting solution is

$$\mathbf{x}(s)=\frac{1}{D(s)}\cdot\mathbf{x}(s)=\frac{1}{s^3-5s^2+s+1}\cdot\begin{bmatrix} s^4+2s^2-6s+1 \\ -2s^3+2 \end{bmatrix}.$$

5. Volterra systems

Direct generalization of convolution systems for nonlinear systems are Volterra systems

$$\begin{aligned} y(j)&=\sum_{l_1=0}^{N-1}h_1(l_1)x(j-l_1)+\sum_{l_1=0}^{N-1}\sum_{l_2=0}^{N-1}h_2(l_1,l_2)x(j-l_1)x(j-l_2)\cdots \\ &=y_1(j)+y_2(j)+\cdots,\quad j=0,1,\ldots,N-1. \end{aligned} \tag{70}$$

Problems connected with Volterra systems can be divided into several items

- determination of Volterra kernels [19]
- calculation of the output of the nonlinear Volterra filter [20]
- determination of inverse kernels to the given nonlinear Volterra system [21].

We shall focus on determination of Volterra kernels. Let us again employ polynomial algebra and express the input signal and $i-th$ Volterra kernel

$$X(z)=\sum_{l=0}^{N-1}x(l)z^l,\quad H_i(z_1,z_2,\cdots,z_i)=\sum_{l_1=0}^{N-1}\sum_{l_2=0}^{N-1}\cdots\sum_{l_i}^{N-1}h_i(z_1,z_2,\cdots,z_i)z_1^{l_1}z_2^{l_2}\cdots z_i^{l_i}.$$

According to [19] for the Volterra kernel of the $i-th$ degree we can write

$$H_i(z_1,z_2,\cdots,z_i)=\frac{Y_i(z_1z_2\cdots z_i)}{X(z_1)X(z_2)\cdots X(z_i)}=\frac{{}^pY_i(z_1,z_2,\cdots,z_i)}{D}. \tag{71}$$

We can again employ the algorithm based on successive reductions of the denominator of (71) (see section 2.2). The algorithm consists in autoconvolutions and crossconvolutions. Let us demonstrate the algorithm using simple example.

Illustrative example:

Let $N=2, i=2$ and

$\mathbf{y}_2 = [3,4]^T$, $\mathbf{x}=[1,2]^T$.

Then according to (71) and after the reductions we have

$$H_2(z_1,z_2) = \frac{3+4z_1z_2}{(1+2z_1)(1+2z_2)} = \frac{(3+4z_1z_2)(1-2z_1)(1-2z_2)}{(1+2z_1)(1-2z_1)(1+2z_2)(1-2z_2)} =$$

$$\frac{19-14z_1-14z_2+16z_1z_2}{(1^2-2^2)^2} = \frac{19-14z_1-14z_2+16z_1z_2}{9}.$$

Hence the resulting kernel of the second degree is

$$h_2 = \frac{1}{9}\begin{bmatrix} 19 & -14 \\ -14 & 16 \end{bmatrix}.$$

6. Conclusions

In the contribution we have presented several error-free algorithms to solve one- and multidimensional convolution systems based on Fermat number theoretical transform. The main significance of the use of modulo arithmetic is eliminating rounding-off and truncation errors.

In the contribution we have derived the error-free algorithm to solve convolution systems that is based on polynomial algebra concept. This form of k-dimensional deconvolution algorithm permits to express it as a sequence of k-dimensional "autoconvolutions" of the response signal and "crossconvolutions" of the response and the output signal.

Further in the contribution we have extended the error-free algorithms to special linear systems, e.g. Toeplitz, Hilbert, Vandermonde systems. We continued with general systems of linear equations. In principle the algorithms proposed in the contribution can be divided into two groups

- iterative (using one modulus – one residue class)
- parallel (using several moduli – several residue classes and Chinese theorem).

The parallel algorithm is well suited for the implementation on parallel computers that allows the increase of the calculation speed.

We have extended the iterative algorithm also for the linear system of polynomial equations. At the end of the contribution we outlined the possible extension of the application of error-free algorithms for nonlinear Volterra systems.

References

[1] N. Ahmed and K.K. Rao, *Orthogonal transforms for digital signal processing*, Springer-Verlag 1975.

[2] H. J. Nussbaumer, *Fast Fourier transform and convolution algorithms*, Springer-Verlag 1981.

[3] M. Morháč, *Precise deconvolution using the Fermat number transform*, *Computers and Mathematics with Applications* **12A** (1986) 319.

[4] M. Morháč, *Error-free deconvolution based on cyclic determinant calculation approach*, *Computer Mathematics* **49** (1993) 41.

[5] M. Morháč, *K-dimensional error-free deconvolution using the Fermat number transform*, *Computers and Mathematics with Applications* **18** (1989) 1023.

[6] M. Morháč, *Precise multidimensional deconvolution using the polynomial algebra concept, International J. Computer Mathematics* **32** (1990) 13.

[7] M. Morháč and V. Matoušek, *Exact algorithm of multidimensional circulant deconvolution, Applied Mathematics and Computation* **164** (2005) 155.

[8] M. Morháč, *An error-free Levinson algorithm to solve integer Toeplitz system, Applied Mathematics and Computation* **61** (1994) 135.

[9] R. E. Blahut, *Fast algorithms for digital signal processing*, IBM Corporation, Owego, NY, 1985.

[10] M. Newman, *Solving equations exactly*, Nat. Bureau Standards 71B:171-179 1967.

[11] M. Morháč, *An algorithm to solve Hilbert systems of linear equations precisely, Applied Mathematics and Computation* **73** (1995) 209.

[12] M. Morháč, *An iterative error-free algorithm to solve Vandermonde systems, Applied Mathematics and Computation* **117** (2001) 45.

[13] W. H. Press et al., *Numerical recipes*, Cambridge University Press, Cambridge, 1986.

[14] M. Morháč, R. Lórencz, *A modular system for solving linear equations exactly. I. Architecture and numerical algorithms, Computers and Artificial Intelligence* **11** (1992) 351.

[15] R. Lórencz, M. Morháč, *A modular system for solving linear equations exactly. II. Hardware realization. Computers and Artificial Intelligence* **11** (1992) 497.

[16] R.T.Gregory, E.V.Krishnamurthy, *Methods and applications of error-free computation*. Springer-Verlag, New York, Berlin, Heidelberg, Tokyo, 1984.

[17] M. Morháč, *One-modulus residue arithmetic algorithm to solve linear equations exactly, Mathematical and Computer Modelling* **19** (1994) 95.

[18] M. Morháč, *An error-free algorithm to solve linear system of polynomial equations, Mathematical and Computer Modelling* **19** (1994) 85.

[19] M. Morháč, *Fast error-free algorithm for the determination of kernels of the periodical Volterra representation, Applied Mathematics and Computation* **38** (1990) 87.

[20] M. Morháč, *A fast algorithm of nonlinear Volterra filtering, IEEE Transactions on Signal Processing* **39** (1991) 2353.

[21] M. Morháč, *Determination of inverse Volterra kernels in nonlinear discrete systems, Nonlinear Analysis, Theory, Methods & Applications* **15** (1990) 269.