Construction of the spatial development model of a city based on vertical planning concepts

Susanne Braun 1, Monika Peters 2, Abram Burnell 3

1 The Hebrew University of Jerusalem
2 Macquarie University
3 Hanoi University of Science and Technology

ABSTRACT

Formation of an urbanistic approach to the study of the urban environment is based on the understanding of how the systems of support and spatial planning can be implemented. The relevance of the study is determined by the fact that urban planning in a traditional way when using horizontal space, may not always affect areas that will ensure expansion and improve the quality of life in cities. The novelty of the study is determined by the fact that vertical planning does not always allow using the general forms and structures of urban planning concepts. The authors have determined that particular importance in planning should be given to the design of the visual environment for the urban area. The paper presents a model for designing a vertical environment and defines the principles of modelling high-rise design concept. It is shown that modelling should be carried out online using real-time technologies. The practical significance of the study is determined by the fact that the construction of the developed model is possible only with the use of spatial modelling technologies on high-resource systems. Obtaining three-dimensional models is assumed in the structure of stereo modelling not only of buildings, but also of accompanying objects that can act as infrastructural assets.

Keywords: Modelling, Planification, Urban planning, Structure, Development.

Corresponding Author:

Susanne Braun
Rachel & Selim Benin School of Computer Science & Engineering
The Hebrew University of Jerusalem
9190401, Derech Brodetski Str., Jerusalem, Israel
E-mail: braun.susann09@gmail.com

1. Introduction

The construction of skyscrapers can be beneficial not only to their owners but also to the authorities and residents of the city, since very often agreements are concluded between the authorities and owners, one of the conditions of which is the additional improvement of the territories adjacent to the future object, in particular the creation of new public spaces [1-12]. The business structure is fundamental in a skyscraper, so everything must be built around it [2; 13; 14]. In this context, apartments located in a skyscraper should not be targeted at everyone, but only at the employees working in this building [3;15-22]. At the same time, all apartments must be rented out (prices will be regulated by the building owners or tenants), without the possibility of purchase and privatization [23-30]. This would, firstly, reduce the burden on the city by reducing commuting and carbon footprint. Second, it will allow the skyscraper to live a fulfilling life without stagnation, as after being fired, employees will vacate apartments, opening those spaces for their successors [31-44].

The problem of unsustainable construction mainly dates back to the 20th century. High-rise buildings consume significantly less energy than low-rise buildings, and modern skyscrapers are more environmentally friendly than all other buildings [45]. In fact, many of them are self-sustaining, which has a beneficial effect on the entire global ecology [46-51]. It also means decentralization and antitrust regulation in this area. This forms an independent city within another city; including in relation to the conditions of cataclysms and epidemics, this is a particularly useful factor. Moreover, sustainability is not the only merit of high-rise
buildings, as skyscrapers are the innovators of many modern technologies in completely different fields of science – from roof gardens, green walls and multi-story greenhouses as part of new vertical forms of urban gardening and agriculture, to carbon fiber technology allowing elevators to safely climb 1000 meters at a time and not burst in the event of a fire [52-63]. Thus, in the future, the skyscraper can be developed as a laboratory for research universities [64]. In this context, skyscrapers not only symbolize the power of new technologies but also use them – even if they symbolize completely different technologies than those used [65; 66; 67]. At the same time, the model that shows the possibility of integrating skyscrapers into the urban environment does not take into account the visual component and the development of a multidimensional research model is relevant. It would allow simulating the general plan of the city online with minimal loss of time.

The use of stereoscopic principles for three-dimensional objects visualization, in the general case, adds complexity in this sense, since, among other things, it is necessary to ensure that all tridimensional development procedures are performed [68-88]. If we use the anaglyphic method of separating stereopairs, when only two mutually complementary colors (red and turquoise) are used, then by reducing the color palette, it is possible to somewhat compensate for the additional difficulties that arise [89-97]. For specialized stereoscopic systems, it is possible to propose a method for speeding up computational procedures based on the idea of using group arithmetic processors in modelling and display [98-102].

2. Materials and methods

A group arithmetic operation should, on the one hand, be dominant in modelling and in computer graphics procedures, and on the other hand, it should allow its implementation on the basis of combinational high-speed structures that require a minimum of control. Previous studies have shown that if the arithmetic operation of scalar multiplication of two real vectors (the sum of paired products) is chosen as the main group operation, then typical problems of linear algebra (matrix-vector operations, systems of linear algebraic equations) admit the effective application of this operation. Many problems of calculating power grids, problems of physics and technology, both directly and when solving differential equations (ordinary and in partial derivatives), are reduced to solving systems of linear algebraic equations of high orders. The results of further research on the subject of accelerating computational procedures based on the indicated group arithmetic operation is presented.

The tasks of modelling and displaying three-dimensional objects require significant computational resources, and often even a powerful CPU cannot meet the modern requirements of the “man-machine” dialogue due to the large volume of three-dimensional graphic information processed during modelling and display [103]. In order to remove most of the computational load associated with manipulating graphic procedures during the display of three-dimensional information and thereby ensure a fast response of the computing environment to a human request, more and more new graphics cards and video accelerators are constantly being developed and put on the market [104]. Modern graphics cards are inherently fast specialised graphics processors designed to perform specific image processing [105]. Display resolutions are constantly increasing; the colour gamut of each screen pixel is constantly growing – therefore, GPUs are becoming more sophisticated and often cost more than a CPU [106-110]. In addition, intensive work is underway to find new algorithms and high-performance computing structures to meet the increased requirements for processing speed and realism of 3D images [111-115].

A system of differential equations is given:

\[
\frac{d\vec{x}}{dt} = A\vec{x} + \vec{F}(t)
\]

where \(A\) – square matrix of constant values; \(\vec{F}(t)\) – vector of the right-hand sides.

To find a solution, the following sequence of actions was proposed: replacing system (1) with a system of the form...
\[
\frac{d\vec{y}}{dt} = A\vec{y} + L_1[\vec{F}(t)]
\]  
\[
\vec{y}(t_0) = \vec{y}_0
\]  

where \( L_1[\vec{F}(t)] \) — linear operator, the application of which to the vector of the right-hand sides \( \vec{F}(t) \) brings system (1) either to a simpler form or to a form for which the answer is known; determination of the vector of variables \( \vec{y}(t) \); finding the required vector \( \vec{x}(t) \) using the so-called inverse operator \( L^{-1}_1[\vec{y}(t)] \), acting by the latter on the vector \( \vec{y}(t) \), that is

\[
\vec{x}(t) = L^{-1}_1[\vec{y}(t)]
\]

Previously, two pairs of linear operators were considered:

first pair,\[ \text{(4)} \]

\[
\begin{align*}
L_{2m}[\vec{F}(t)] &= \frac{d^m}{dt^m} [\vec{F}(t)] \\
L^{-1}_{2m}[\vec{y}(t)] &= \int_{t_0}^{t} \int_{t_0}^{t} \ldots \int_{t_0}^{t} \frac{d\tau_1}{m} \ldots \frac{d\tau_m}{m} \\
\text{with constraints} & \\
\begin{cases}
\vec{F}(t_0) = 0 \\
\frac{d^l[\vec{F}(t)]}{dt^l} \big|_{t = t_0} = 0 \\
l = 1,2,\ldots,m - 1
\end{cases}
\]

The only constraints of the second pair (6)

\[
\begin{align*}
L_{2m}[\vec{F}(t)] &= \int_{t_0}^{t} \int_{t_0}^{t} \ldots \int_{t_0}^{t} \vec{F}(t) \frac{d\tau_1}{m} \ldots \frac{d\tau_m}{m} \\
L^{-1}_{2m}[\vec{y}(t)] &= \frac{d^m}{dt^m} [\vec{y}(t)] \\
\text{with the conditions of vector integration} & \\
\int_{t_0}^{t} \vec{F}(t) d\tau, \int_{t_0}^{t} \int_{t_0}^{t} \vec{F}(t) d\tau d\tau
\end{align*}
\]

3. Results and discussions

It is clear that it is expedient to use these pairs of operators only in those cases when the components of the vector \( \vec{F}(t) \) are the same, since the operators act identically on all components of the vectors \( \vec{F}(t) \) and \( \vec{y}(t) \). This narrows down the class of these systems. In order to expand this class, it is proposed to use operator matrices instead of pairs of linear operators. In this case, instead of direct linear operators the vector
\( \vec{y}(t) \) will be acted upon by the diagonal matrix of linear operators \( L \), in which the direct operators \( L_{1m}, L_{2m} \) can stand on the diagonal in any combination and in any order:

\[
L = \begin{bmatrix}
L_{1m} & 0 & 0 & 0 & 0 \\
0 & L_{2k} & 0 & 0 & 0 \\
0 & 0 & L_{2l} & 0 & 0 \\
0 & 0 & 0 & \ddots & 0 \\
0 & 0 & 0 & 0 & L_{1f}
\end{bmatrix}
\]

(7)

The vector \( \vec{y}(t) \) is acted upon by the diagonal matrix of inverse linear operators \( L^{-1} \):

\[
L^{-1} = \begin{bmatrix}
L_{1m}^{-1} & 0 & 0 & 0 & 0 \\
0 & L_{2k}^{-1} & 0 & 0 & 0 \\
0 & 0 & L_{2l}^{-1} & 0 & 0 \\
0 & 0 & 0 & \ddots & 0 \\
0 & 0 & 0 & 0 & L_{1f}^{-1}
\end{bmatrix}
\]

(8)

It is convenient to prove the possibility of using the proposed matrices of inverse operators in the Laplace image domain. Suppose that the vector \( \vec{F}(t) \) of system (1) under zero initial conditions is acted upon by the matrix of operators (7) under constraints (5) on all necessary operators. Then,

\[
\frac{d\vec{y}}{dt} = A\vec{y} + L[\vec{F}(t)]
\]

(9)

Apply the Laplace transform

\[
p\vec{Y}(p) = A\vec{Y}(p) + L(p)\vec{F}(p)
\]

(10)

where

\[
L(p) = \begin{bmatrix}
p^{\pm i} & 0 & 0 & 0 \\
0 & p^{\pm j} & 0 & 0 \\
0 & 0 & \ddots & 0 \\
0 & 0 & 0 & p^{\pm k}
\end{bmatrix}
\]

(11)

From where

\[
\vec{Y}(p) = (pE - A)^{-1}L(p)\vec{F}(p)
\]

(12)

In the operator domain, the influence of the matrix of operators on the vector \( \vec{F}(t) \) is reduced to the multiplication of matrix (11) by the vector \( \vec{F}(p) \). The required vector of unknowns of system (1) in the image domain, on the one hand, equals
On the other hand, applying a matrix of inverse operators in the image domain to a vector gives

\[ L^{-1}(p) \hat{Y}(p) = L^{-1}(p)(pE - A)^{-1}L(p) \hat{F}(p) \]  

(14)

where

\[ L^{-1}(p) = \begin{bmatrix} p^{\pm i} & 0 & 0 & 0 \\ 0 & p^{\pm j} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & p^{\pm k} \end{bmatrix} \]  

(15)

Since the matrix \( L^{-1}(p) \) is diagonal, expression (14) can be transformed to the form

\[ L^{-1}(p) \hat{Y}(p) = (pE - A)^{-1}L^{-1}(p)L(p) \hat{F}(p) = (pE - A)^{-1}E \hat{F}(p) = \hat{X}(p) \]  

(16)

which it was necessary to prove.

The fact that the Laplace image domain:

\[ L^{-1}(p)L(p) = E, \]  

(17)

served as the fact that it was accepted to write the matrix of inverse operators \( L^{-1}(p) \), corresponding to the spelling adopted for the image of the inverse matrix. Formally, in the time domain, the action of the matrix of direct operators on the vector \( \hat{F}(t) \) is reduced to three steps: multiplying the components of the matrix \( L \) by the vector \( \hat{F}(t) \); introducing each component of the vector \( \hat{F}(t) \) under the sign of the corresponding linear direct operator; performing actions provided by direct operators. Similarly, from the formal standpoint, the actions of the matrix of inverse operators \( L^{-1}(p) \) over the vector \( \hat{y}(t) \) are performed:

1) components of the matrix \( L^{-1}(p) \) multiplied by the vector \( \hat{y}(t) \);
2) each component of the vector \( \hat{y}(t) \) is introduced under the sign of the corresponding linear inverse operator; the actions provided by the inverse operators are performed.

Example. Given the system:

\[ \begin{align*}
\frac{dx_1}{dt} & = (a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4) = b_1t^5 \\
\frac{dx_2}{dt} & = (a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4) = b_2t^4 \\
\frac{dx_3}{dt} & = (a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4) = b_3t^2 \\
\frac{dx_4}{dt} & = (a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4) = b_4t^{-3} \\
\end{align*} \]

\[ (x_1, x_2, x_3, x_4)|_{t=0} = 0 \]  

(18)
It is necessary to determine the vector $x(t)$ provided that the answer for the case

$$\begin{cases}
\frac{d\vec{y}}{dt} = A\vec{y} + \vec{\varphi}(t) \\
\vec{y}(t_0) = \vec{y}_0
\end{cases}$$

(19)

Where

$$\varphi_1(t) = c_1,$$

$$\varphi_2(t) = c_2t^{-1},$$

$$\varphi_3(t) = c_3,$$

$$\varphi_4(t) = c_4t^{-1}.$$  

(20), (21), (22), (23)

$c_1, c_2, c_3, c_4$ – known constants.

In this case, it is advisable to choose the matrix of direct linear operators of the form:

$$L(p) = \begin{bmatrix}
    p^5 & 0 & 0 & 0 \\
    0 & p^{-2} & 0 & 0 \\
    0 & 0 & p^3 & 0 \\
    0 & 0 & 0 & p^{-2}
\end{bmatrix}$$

(24)

which corresponds to the matrix in the time domain

$$L = \begin{bmatrix}
\frac{d^5}{dt^5} [0] & 0 & 0 & 0 \\
0 & \int_{t_0}^{t} \int_{t_0}^{t} F_1(t) dt d\tau & 0 & 0 \\
0 & 0 & \frac{d^3}{dt^3} & 0 \\
0 & 0 & 0 & \int_{t_0}^{t} \int_{t_0}^{t} F_1(t) dt d\tau
\end{bmatrix}$$

(25)

Step 1.

$$L[F_1(t)] = \begin{bmatrix}
L_{15} & 0 & 0 & 0 \\
0 & L_{22} & 0 & 0 \\
0 & 0 & L_{13} & 0 \\
0 & 0 & 0 & L_{22}
\end{bmatrix} \begin{bmatrix}
F_1(t) \\
F_2(t) \\
F_3(t) \\
F_4(t)
\end{bmatrix} = \begin{bmatrix}
L_{15}F_1(t) \\
L_{22}F_2(t) \\
L_{13}F_3(t) \\
L_{22}F_4(t)
\end{bmatrix}$$

(26)

Step 2.
Step 3.

\[
\begin{aligned}
L_{14}[F_1(t)] &= \frac{d^5}{dt^5} [F_1(t)] = \frac{d^5}{dt^5} [b_t t^5] \\
L_{23}[F_2(t)] &= \int_{t_0}^{t} \int_{t_0}^{\tau} \int_{t_0}^{\tau} F_2(\tau) d\tau d\tau d\tau = \int_{t_0}^{t} \int_{t_0}^{\tau} b_2 \tau^{-4} d\tau d\tau d\tau \\
L_{13}[F_3(t)] &= \frac{d^3}{dt^3} [F_3(t)] = \frac{d^3}{dt^3} [b_3 t^3] \\
L_{22}[F_4(t)] &= \int_{t_0}^{t} \int_{t_0}^{\tau} F_4(\tau) d\tau d\tau = \int_{t_0}^{t} \int_{t_0}^{\tau} b_4 \tau^{-3} d\tau d\tau \\
\end{aligned}
\]

(27)

Constants (20) make it possible to determine the vector \( \tilde{y}(t) \) from the system (19). Then, to obtain the required answer, the matrix of inverse operators is applied to the vector \( \tilde{y}(t) \), which in this case takes the form:

Reverse step 1.

\[
L^{-1} = \begin{bmatrix}
\frac{1}{c_1} \int_{0}^{t} d\tau & \cdots & \int_{0}^{t} d\tau & \cdots & \int_{0}^{t} d\tau \\
\frac{5}{5} & 0 & 0 & 0 & 0 \\
0 & \frac{1}{c_2} \frac{d^2}{dt^2} & 0 & 0 & 0 \\
0 & 0 & \frac{1}{c_3} \int_{0}^{t} d\tau & \cdots & \int_{0}^{t} d\tau \\
0 & 0 & 0 & \frac{3}{3} & \frac{1}{c_4} \frac{d^3}{dt^3} \\
\end{bmatrix}
\]

(29)

or in the image domain:

\[
L^{-1}(p) = \begin{bmatrix}
\frac{1}{c_1} p^{-5} & 0 & 0 & 0 \\
0 & \frac{1}{c_2} p^3 & 0 & 0 \\
0 & 0 & \frac{1}{c_3} p^{-3} & 0 \\
0 & 0 & 0 & \frac{1}{c_4} p^3 \\
\end{bmatrix}
\]

(30)

Reverse step 2.
Reverse step 3, as indicated above, is to implement the mathematical operations provided by the inverse operators. An analysis of the formulas obtained by the T-transformation method as applied to differential equations shows that most of the computational load falls on the group arithmetic operation – the sum of paired products. On the other hand, it is known that such an operation is effectively implemented in a non-positional number in residue number system (RS), since it consists only of modular operations of addition and multiplication. To increase the efficiency of solving differential equations, the idea arose to combine these two approaches.

System of linear differential equations with constant coefficients, Cauchy problem. The system (32)

\[
\begin{aligned}
\frac{d\vec{x}}{dt} &= \bar{A}\vec{x} + \vec{f}(t) \\
\vec{x}_{t=\tau_0} &= \vec{x}_0 
\end{aligned}
\]  

(32)

where \(\vec{x}\) – vector of unknowns; \(\vec{f}(t)\) – vector of the right-hand sides; \(\vec{x}_0\) – vector of initial values; \(\bar{A}\) – matrix of constant coefficients.

In the region of T-transformations, system (32) takes the form

\[
D\vec{x}(k) = \bar{A}\vec{x}(k) + \vec{F}(k)
\]  

(33)

where \(D\) – symbol of the Taylor derivative, which in this case applies to each component of the vector \(\vec{x}(k); \vec{x}(k), \vec{F}(k)\) – corresponding images of the vectors \(\vec{x}(t)\) and \(\vec{f}(t)\); \(k\) – discrete argument taking on values 0,1,2, ...

After the implementation of the \(T\)-derivative, the calculation formula will take the form:

\[
\vec{x}(k+1) = \frac{H}{k+1} \left[ A\vec{x}(k) + \vec{F}(k) \right]
\]  

(33)

where \(H\) – constant with the dimension of time \(t\).

The main computational load in formula (33) falls on the operation \(A\vec{x}(k)\), that is, on a finite number of sums of paired products. However, it is not possible to fully implement formula (19) on computational structures operating in the residue system, because at each step of searching for the next discrete vector of unknowns \(\vec{x}_i\), it is necessary to perform the operation of dividing by a value that is so laborious in a non-positional number system that the advantage that residue system gives when performing modular operations.
Therefore, the task arose – to transform formula (33) so that the calculation of the further vector of unknowns (or a function of the vector) occurs without using the division operation over the vector (or its function) at the previous step. It was proposed to solve this problem by introducing a new vector of variables,

\[ \tilde{y}(k) = \varphi(k) \tilde{x}(k) \]  

(34)

where \( \varphi(k) \) – function of an integer argument (meaning that its calculation does not require non-modular operations).

Substitute (34) in (33):

\[ \frac{\tilde{y}(k+1)}{\varphi(k+1)} = \frac{H}{k+1} \left[ A \tilde{y}(k) + \tilde{F}(k) \right] \]  

(35)

After transformation we obtain:

\[ \frac{\tilde{y}(k+1)}{\varphi(k+1)} = \frac{H}{k+1} \frac{1}{\varphi(k)} \left[ A \tilde{y}(k) + \varphi(k) \tilde{F}(k) \right] \]  

(36)

From formula (36) it follows that to satisfy the above conditions, the function \( \varphi(k) \) must have the following property:

\[ \varphi(k + 1) = (k + 1) \varphi(k) \]  

(37)

Indeed, if condition (37) is satisfied, then formula (36) is transformed into the form (38):

\[ \tilde{y}[k + 1] = H \left[ A \tilde{y}(k) + \varphi(k) \tilde{F}(k) \right] \]  

(38)

from which it follows that to calculate any subsequent vector of unknowns \( \tilde{y}(k + 1) \), only modular operations are needed, that is, the results of their execution remain in the class of integers if there were initial data in this class. The function \( \varphi(k) \) that simultaneously satisfies both formula (37) and the conditions mentioned above (41) is the factorial of the variable:

\[ \varphi(k) = k! \]  

(39)

The validity of this statement is confirmed, first, by the fact that for calculating \( k! \) only the operation of multiplication is needed, and secondly, by the fact that

\[ (k + 1)! = (k + 1)k! \]  

(40)

Satisfies (43).

Finally, formulas (41) and (42) have the form:

\[ \tilde{y}(k) = k! \tilde{x}(k) \]  

(41)
\[ \vec{y}[k + 1] = H[\vec{A}\vec{y}(k) + k1\vec{F}(k)] \] (42)

The opinion is erroneous that the change of the vector of variables (41) does not give anything significant on the grounds that we are ultimately interested in the vector \( \vec{x}(k) \), and not in the vector \( \vec{y}(k) \). Note, however, that the transition to the vector of variables \( \vec{y}(k) \) makes it possible to combine non-modular division operations \( \frac{\vec{y}(k)}{k!} \) and modular operations for calculating the next discrete vector \( \vec{y}(k + 1) \). Moreover, it is important that by the time the next discrete vector \( \vec{y}(k + 1) \) is calculated, all the necessary components in the residue system (RS) are already available.

The original formula (33) does not provide such possibility. Indeed, suppose that the numerator of formula (33) is calculated on the RS basis; and the operation of division by \( (k + 1)! \), is associated with the calculation of the vector of unknowns \( \vec{x}(k + 1) \), similar to the case considered above, which is performed in the positional number system. All this is possible, but during the transformation of the numerator of formula (33) from the RS into a positional basis and during the further execution of the operation of division by \( (k + 1)! \), and then during the transformation of the vector \( \vec{x}(k + 1) \) from the positional form into the RS – non-positional blocks are in the waiting state, since to calculate the discrete vector of the numerator of formula (33) at the next step, the vector \( \vec{x}(i + 1) \) is needed, moreover, in a non-positional basis.

Returning to the new formulas (41), (42), the following distribution of computational functions between positional and non-positional blocks is proposed. A specialised processor of the group operation, working in the RS, is busy calculating the discrete vector by the formula (42). Calculation of the discrete vector \( \vec{x}(k + 1) \), checking for accuracy, as well as obtaining points of the required vector of a given differential equation (43) is performed in blocks with a positional basis,

\[ \vec{x}(i + 1) = \vec{y}_i(0) + \vec{y}_i(1) + \vec{y}_i(2) + \frac{\vec{y}_i(3)}{2!} + \ldots \] (43)

Moreover, as noted above, at the moment of calculating by the non-positional processor, for example, the vector \( \vec{y}_i(4) \), the transformation from the RS to the positional basis of the previously obtained value \( \vec{y}_i(3) \) occurs in parallel; the vector \( \vec{x}_i(4) \) in the positional number system; the desired result is accumulated in positional memory by summing \( \vec{y}_i(0) + \vec{y}_i(1) + \frac{\vec{y}_i(2)}{2!} + \ldots \). When calculating using formula (26), the components of the vector \( \vec{y}(k + 1), HA\vec{y}(k), Hk1\vec{F}(k) \), should be stored separately in memory, because when performing the next division operation \( \frac{\vec{y}(k + 1)}{(k + 1)!} \), the range of representation of numbers for separate division is less than for the division of the entire vector by \( (k + 1)! \). In other words, it is necessary to transform the value \( HA\vec{y}(k) \) from RS into positional form and here to determine the first component.
Second component:

\[
\frac{H_{k+1}\hat{F}(k)}{(k+1)!} = \frac{H\hat{F}(k)}{k+1}
\]  

(44)

must be fully calculated in positional form, since there is all the data for this operation in positional memory. Further, it remains only to add the resulting components of the vector \( \hat{y}(k+1) \).

System of linear differential equations with variable coefficients, the Cauchy problem. The system (67)

\[
\begin{cases}
\frac{d\hat{x}}{dt} = A(t)\hat{x} + \hat{f}(t) \\
\hat{x}|_{t=t_0} = \hat{x}_0
\end{cases}
\]

(45)

where the coefficients of the matrix \( A(t) \) depend on time. According to the technique, system (67) in the region of \( T \)-images is represented as follows:

\[
D\hat{x}(k) = A(k) \times \hat{x}(k) + \hat{F}(k)
\]

(46)

where \( A(k) \times \hat{x}(k) \) is the product in the \( T \)-domain.

The calculation formula after the implementation of the \( T \)-derivative and \( T \)-product has the form:

\[
\hat{x}(k+1) = \frac{H}{k+1} \left[ \sum_{l=0}^{k} A(l)\hat{x}(k-l) + \hat{F}(k) \right]
\]

(47)

As can be seen from formula (47), the main complication in setting the system of differential equations (48) on computational structures with mixed coding is the presence of the division by the coefficient \( \frac{H}{k+1} \) at each step of calculating the discrete vector \( \hat{x}(k+1) \).

Let us try to use the change of variables found earlier (25). The proof will be carried out by the method of mathematical induction. First, let us verify the efficiency of substitution (41) in (47) for the cases when \( k = 0, 1, 2 \):

\( k = 0 \):

\[
\hat{x}(1) = H[A(0)\hat{x}(0) + \hat{F}(0)] = \hat{y}(1)
\]

(48)

division operation is not necessary:

\( k = 1 \):

\[
\hat{y}(1) = H[A(0)\hat{y}(0) + \hat{F}(0)]
\]

(49)

\[
\hat{x}(2) = \frac{H}{2} [A(0)\hat{x}(1) + A(2)\hat{x}(0) + \hat{F}(1)] = \frac{\hat{y}(2)}{2!}
\]

(50)
division operation is not necessary:

\[ \hat{y}(2) = H[A(0)\hat{y}(1) + A(1)\hat{y}(0) + \hat{F}(1)] \]  
(51)

\[ \hat{x}(3) = \frac{H}{3} [A(0)\hat{x}(2) + A(1)\hat{x}(1) + A(2)\hat{x}(0) + \hat{F}(2)] = \frac{H}{3} [A(0)\frac{\hat{y}(2)}{2!} + A(1)\hat{y}(1) + A(2)\hat{y}(0) + \hat{F}(2)] = \frac{H}{3!} [A(0)\hat{y}(2) + 2!A(1)\hat{y}(1) + 2!A(2)\hat{y}(0) + 2!\hat{F}(2)] = \frac{\hat{y}(3)}{3!} \]  
(52)

division operation is not necessary:

\[ \hat{y}(3) = H[A(0)\hat{y}(2) + 2!A(1)\hat{y}(1) + 2!A(2)\hat{y}(0) + 2!\hat{F}(2)] \]  
(53)

\[ \hat{x}(4) = \frac{H}{4} [A(0)\hat{x}(3) + A(1)\hat{x}(2) + A(2)\hat{x}(1) + A(3)\hat{x}(0) + \hat{F}(3)] = \frac{H}{4} [A(0)\frac{\hat{y}(3)}{3!} + A(1)\frac{\hat{y}(2)}{2!} + A(2)\hat{y}(1) + A(3)\hat{y}(0) + 3!\hat{F}(3)] = \frac{\hat{y}(4)}{4!} \]  
(54)

division operation is not necessary:

\[ \hat{y}(4) = H[A(0)\hat{y}(3) + 3!A(1)\hat{y}(2) + 3!A(2)\hat{y}(1) + 3!A(3)\hat{y}(0) + 3!\hat{F}(3)] \]  
(55)

Suppose that the above procedure is also valid for the case when \( k = n \), that is, the vector \( \hat{y}(n) \) is determined excluding the division operation, and under this condition, consider the case:

\[ k = n + 1 \]  
(56)

\[ \hat{x}(n + 1) = \frac{H}{n+1} [A(0)\hat{x}(n) + A(1)\hat{x}(n-1) + A(2)\hat{x}(n-2) + \ldots + A(n-1)\hat{x}(1) + A(n)\hat{x}(0) + \hat{F}(n)] \]  
(57)

Substituting (41) into (53), obtain:

\[ \hat{x}(n + 1) = \frac{H}{n+1} \left[ A(0)\frac{\hat{y}(n)}{n!} + A(1)\frac{\hat{y}(n-1)}{(n-1)!} + A(2)\frac{\hat{y}(n-2)}{(n-2)!} + \ldots + A(n-1)\hat{y}(1) + A(n)\hat{y}(0) + \hat{F}(n) \right] \]  
(58)

After conversion:
or more compact form:

\[
\tilde{x}(n + 1) = \frac{H}{(n + 1)!} \left[ A(0)\tilde{y}(n) + nA(1)\tilde{y}(n - 1) + (n - 1)A(2)\tilde{y}(n - 2) + \cdots + n!A(n - 1)\tilde{y}(1) + n!A(n)\tilde{y}(0) + n!\tilde{F}(n) \right]
\]  

(59)

Substituting (41) into (60), obtain:

\[
\tilde{y}(n + 1) = H\left[ A(0)\tilde{y}(n) + \sum_{l=1}^{n} A(l)\tilde{y}(n - l) \prod_{i=l}^{n-1} (n - i) + n!\tilde{F}(n) \right]
\]  

(61)

As can be seen, the vector \( \tilde{y}(n + 1) \) can be calculated without the division operation. By induction, the above procedure is valid for any values of the integer argument \( k \).

Thus, the proposed change of variables (41), applied to the calculation formula (53), makes it possible to parallelise the computation process in blocks operating in the positional and non-positional number system.

As in the system of linear differential equations with constant coefficients, calculations related to the vector \( \tilde{y}(k) \), formula (61), it is advisable to perform in a non-positional basis (on a special processor of a group operation). In addition, specialised structures operating in the residue system (RS) can be effective in representing the value of \( k! \). Calculations related to the implementation of the division operation are performed in a positional basis.

Approximate methods for solving systems of nonlinear equations requires a large number of arithmetic operations of the same type. Therefore, the computational process corresponding to this problem, when the latter is set up on universal computers, takes a lot of computer time. The carried-out elaboration of problems of linear algebra and analysis prove the efficiency of the use of group arithmetic operation of the dot product of two real vectors. Below, an analysis will be carried out for the application of a group operation to speed up the computational process when solving systems of nonlinear equations.

Newton method. A system of nonlinear equations with real left-hand sides is given:

\[
\begin{align*}
    f_1(x_1, x_2, \ldots, x_n) &= 0 \\
    f_2(x_1, x_2, \ldots, x_n) &= 0 \\
    \cdots & \cdots \cdots \\
    f_n(x_1, x_2, \ldots, x_n) &= 0 \\
\end{align*}
\]

(62)

Or \( \tilde{F}(\tilde{x}) = 0 \), where \( \tilde{x} \) – vector of unknowns, \( \tilde{F}(\tilde{x}) \) – vector function.

Assuming that the function \( \tilde{F}(\tilde{x}) \) is continuously differentiated in some convex domain that contains:

\[
\tilde{x}^{[k]} = \left( x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)} \right)
\]

(63)

it is proved that the iterative process of finding the required vector can be organised by the formula:
\[
\hat{x}^{(k+1)} = \hat{x}^{(k)} - W^{-1}(\hat{x}^{(k)}) \tilde{F}(\hat{x}^{(k)}), k = 0,1,2,\ldots
\] (64)

where \( W^{-1}(\hat{x}^{(k)}) \) – the inverse Jacobi matrix of the system of functions \( f_1, f_2, \ldots, f_n \) with respect to the variables \( x_1, x_2, \ldots, x_n \) for \( k \)-th approximation,

\[
W^{-1}(\hat{x}^{(k)}) = \begin{bmatrix}
\frac{df_1}{dx_1^{(k)}} & \frac{df_2}{dx_2^{(k)}} & \cdots & \frac{df_n}{dx_n^{(k)}} \\
\frac{df_1}{dx_2^{(k)}} & \frac{df_2}{dx_2^{(k)}} & \cdots & \frac{df_n}{dx_n^{(k)}} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{df_1}{dx_n^{(k)}} & \frac{df_2}{dx_n^{(k)}} & \cdots & \frac{df_n}{dx_n^{(k)}}
\end{bmatrix}^{-1}
\] (65)

From the standpoint of the computational procedure complexity, the inconvenience of Newton method is the need to calculate the inverse matrix \( W^{-1}(\hat{x}^{(k)}) \) at each step. It is proved that if the matrix \( \hat{x}^{(0)} \) is continuous in proximity to the required solution \( \hat{x}^{(0)} \), then approximately assuming the solution and the initial approximation:

\[
W^{-1}(\hat{x}^{(k)}) \approx W^{-1}(\hat{x}^{(0)})
\] (66)

come to a modified Newton process:

\[
\tilde{y}^{(k+1)} = \tilde{y}^{(k)} - W^{-1}(\tilde{y}^{(0)}) \tilde{F}(\tilde{y}^{(0)}), k = 0,1,2,\ldots, \tilde{y}^{(0)} = \hat{x}^{(0)}
\] (67)

In this case, the inverse matrix \( W^{-1}(\hat{x}^{(0)}) \) needs to be calculated only once during the task execution time; therefore, this work and the calculations of the vector \( \tilde{F}(\tilde{y}^{(0)}) \) can be assigned to the CPU; the rest of the cumbersome computations of the same type, associated with multiplying a matrix by a column at each iteration step, will be effectively performed by a special processor of the above group arithmetic operation.

Denoting

\[
W^{-1}(\hat{x}^{(0)}) = G
\] (68)

\[
G = \begin{bmatrix}
g_{11} & g_{11} & \cdots & g_{1n} \\
g_{21} & g_{12} & \cdots & g_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
g_{n1} & g_{n2} & \cdots & g_{nn}
\end{bmatrix}
\] (69)

and representing (67) by component, the formulas consisting only of the sums of even products are obtained:
If it is necessary to apply the unmodified Newton method (66), the operation of finding the elements of the inverse matrix in terms of complexity comes to the fore. From the standpoint of using the group arithmetic operation, the most suitable of the general methods for finding the inverse matrix are the border method and the completion method. Border method. The idea of the method is that the original \( n \)-th order matrix \( A_n \) is considered as the result of bordering the matrix \( A_{n-1}(n-1) \)-th order:

\[
A = A_n = \begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n-1} & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n-1} & a_{2n} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
a_{n-11} & a_{n-12} & \cdots & a_{n-1n-1} & a_{n-1n} \\
a_{n1} & a_{n2} & \cdots & a_{nn-1} & a_{nn}
\end{bmatrix} = \begin{bmatrix}
\vec{v}_n \\
\vec{u}_n
\end{bmatrix}
\]

where

\[
\vec{v}_n = (a_{n1}, \ldots, a_{nn-1}),
\]

\[
\vec{u}_n = (a_{1n}, \ldots, a_{n-1n}),
\]

The matrix \( A^{-1} \) is also sought in the form of a bordered matrix. It has been proven that:

\[
A^{-1} = \begin{bmatrix}
A_{n-1}^{-1} + \frac{A_{n-1}^{-1} \vec{v}_n \vec{u}_n^T A_{n-1}^{-1}}{\alpha_n} \\
-\frac{\vec{v}_n \vec{u}_n^T A_{n-1}^{-1}}{\alpha_n} \\
\frac{1}{\alpha_n}
\end{bmatrix}
\]

where

\[
\alpha_n = \vec{v}_n A_{n-1}^{-1} \vec{u}_n.
\]

Formula (75) is the basis for the sequential border method. Inverse matrices are sequentially constructed for matrices:

\[
\begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{bmatrix}
\]

\[
\begin{bmatrix}
\alpha_{11} & \alpha_{12} & \alpha_{13} \\
\alpha_{21} & \alpha_{22} & \alpha_{23} \\
\alpha_{31} & \alpha_{32} & \alpha_{33}
\end{bmatrix}
\]

(76)
of which each subsequent one leaves the previous one with the help of a border on the basis of formula (68).

This process combines the following actions (provided that \( A_{n-1}^{-1} \) is already known, the elements of the matrix \( A_{n-1}^{-1} \) are sought:

1. Column \( \tilde{A}_{n-1}^{-1} \) is calculated.
2. String \( \tilde{v} \) with elements \( y_{n1}, \ldots, y_{nn-1} \) is calculated.
3. The number is calculated

\[
\alpha_n = \alpha_{nn} + \sum_{i=1}^{n-1} \alpha_{ni} \beta_{in} = \alpha_{nn} + \sum_{i=1}^{n-1} \alpha_{in} y_{ni}
\]  

(77)

4. The elements \( d_{ik} \) of the inverse matrix are calculated by the formulas:

\[
d_{ik} = d_{ik}' + \frac{\beta_{in} y_{nk}}{\sigma_n}, (i, k \leq n - 1)
\]  

(78)

\[
d_{in} = \frac{\beta_{in}}{\sigma_n}
\]  

(79)

\[
d_{nk} = \frac{y_{nk}}{\sigma_n}, (i, k \leq n - 1).
\]  

(80)

\[
d_{nn} = \frac{1}{\alpha_n}
\]  

(81)

where \( d_{ik}' \) – elements of the matrix \( A_{n-1}^{-1} \).

Calculations by formulas (81) can be successfully performed on a special processor of a group operation. It is advisable to assign the implementation of dependencies by the item 4 to the CPU.

Replenishment method. The idea of the method is as follows, let \( B \) be a non-personal matrix for which the inverse is known. It is necessary to determine the inverse matrix \( A^{-1} \), related to the matrix \( B \) by the formula

\[
A = B + \tilde{u} \tilde{v}
\]  

(82)

where \( \tilde{u} \) – some well-known column; \( \tilde{v} \) – some known string:

\[
\tilde{u} \tilde{v} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \circ \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & \cdots & u_1 v_n \\ u_2 v_1 & u_2 v_2 & \cdots & u_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_n v_1 & u_n v_2 & \cdots & u_n v_n \end{bmatrix}
\]  

(83)

It is proved that the inverse matrix

\[
A^{-1} = B^{-1} - \frac{1}{\gamma} B^{-1} \tilde{u} \tilde{v} B^{-1}
\]  

(84)

where
\[ y = 1 + \mathbf{v} B^{-1}, \quad (85) \]

This idea is usually extended to the special case when the matrix \( A \) leaves the matrix \( B \) by changing one string, that is

\[ A = B + V \quad (86) \]

where \( V \) — matrix, all elements of which are equal to zero, except for elements of a variable string, for example, with \( k \). Then,

\[ V = \mathbf{u} \mathbf{v} = \varepsilon_k \mathbf{v}, \quad (87) \]

where \( \mathbf{v} \) — nonzero strings of the matrix \( V \); \( \varepsilon_k \) — column, \( k \)-th element of which is equal to one, and the rest is zero. In this case,

\[ \tilde{\alpha}_j = \tilde{\beta}_j - \frac{\langle \mathbf{v}', \tilde{\beta}_j \rangle}{1 + \langle \mathbf{v}, \tilde{\alpha}_k \rangle} \tilde{\beta}_k \quad (88) \]

where \( \tilde{\mathbf{B}} \) — \( k \)-th column of the matrix \( B^{-1} \); \( \tilde{\alpha}_j \) — \( j \)-th column of the matrix \( \alpha^{-1} \); \( \langle \mathbf{v}', \tilde{\alpha}_j \rangle \) — scalar products of vectors.

The question arises of where to take for an arbitrary matrix \( A \) the matrix \( B \) satisfying formula (70), for which the inverse is known. The completion method provides that a given matrix \( A \) is considered as the last member of the sequence \( A_0, A_1, A_2, \ldots, A_n = A_0 A_0 = E \) (identity matrix).

The transition from the previous matrix \( A_{k-1} \) to the next \( A_k \) is carried out by replacing the \( k \)-th string of the matrix \( A_{k-1} \) with the \( k \)-th string of the matrix \( A_k \) with \( n \) iterations.

The transition formulas at the \( k \)-th step are as follows:

\[ \tilde{\alpha}_j^{(k)} = \tilde{\alpha}_j^{(k-1)} - \frac{\langle \mathbf{v}', \tilde{\alpha}_j \rangle^{(k-1)}}{1 + \langle \mathbf{v}, \tilde{\alpha}_j \rangle^{(k-1)}} \tilde{\alpha}_k^{(k-1)} \quad (89) \]

where \( \tilde{\alpha}_j^{(k)} \) — \( j \)-th column of the matrix

\[ A_k^{-1} \mathbf{v} = (a_{k1}, a_{k2}, \ldots, a_{kn}). \quad (90) \]

Operations in brackets of formula (90) are the scalar product of two vectors, which are expedient perform group operations on the processor. The operation of finding the elements of the inverse matrix can be accelerated if \( A^{-1} \). It makes sense for the first calculations to be performed with
rounding by Newton method. In the future, as the solution is approached, it may be necessary to refine the value of the inverse Jacobian – more accurately find the elements of the inverse matrix. The use of a special processor of the group operation will be effective in this case as well. Indeed, first, we will effectively obtain the control relation

\[ A \tilde{A}^{-1} \]

, to find which \( n^2 \) is necessary so the norm of the indicated group operations if the relation

\[ \| E - A \tilde{A}^{-1} \| \leq K < 1 \]  

(91)

where \( \| -A \tilde{A}^{-1} \| \) – the first or second norm of the matrix, then in this case the elements of the inverse matrix \( A^{-1} \) can be calculated with an arbitrarily high accuracy using an iterative process:

\[
\begin{cases}
    A_1^{-1} = A_0^{-1}(E + R_0), R_1 = E - A A^{-1}_1, \\
    A_2^{-1} = A_1^{-1}(E + R_1), R_2 = E - A A^{-1}_2, \\
    \vdots \\
    A_m^{-1} = A_{m-1}^{-1}(E + Rm - 1), R_m = E - A A_{m-1}^{-1}
\end{cases}
\]  

(92)

It is proved that if condition (40) is met, the convergence of the process is very fast – the number of correct decimal places grows exponentially. Secondly, the form of formulas (92) convincingly testifies to the efficiency of using the processor of sums of paired products, since in these formulas the most laborious and preferable operation is the multiplication of two square matrices, which requires \( n^2 \) of the indicated group operations.

Fastest descent method. Assuming that the functions \( f_1, f_2, ..., f_n \) of formulas (65) are real and continuously differentiable in their common domain of definition, it is proved that the iterative process of the fastest descent that converges is performed as follows:

\[ \tilde{x}^{(k+1)} = \tilde{x}^{(k)} - \mu_k W_k \tilde{F}(\tilde{x}) , k = 0, 1, 2, ... \]  

(93)

where

\[ \tilde{F} = \tilde{F}(\tilde{x}^{(k)}) \]  

(94)

\[ W_k = W(\tilde{x}^{(k)}) \]  

(92)

Jacobi matrix:

\[ W(\tilde{x}) = \frac{\partial \tilde{F}}{\partial \tilde{x}} \]  

(95)

\( W'(\tilde{x}) \) – transposed matrix
\[
\mu^{(k)} = \left( \frac{\vec{\xi}^{(k)} \cdot \vec{W} \cdot \vec{\xi}^{(k)}}{\vec{W} \cdot \vec{\xi}^{(k)} \cdot \vec{W} \cdot \vec{\xi}^{(k)} + \vec{\xi}^{(k)} \cdot \vec{\xi}^{(k)}} \right)
\]  

(introduction the notation)

\[
W \cdot \vec{\xi}^{(k)} = \vec{\xi}^{(k)}
\]  

Represent the matrix \( W \) in another way:

\[
W = \begin{bmatrix}
  w_{11} & w_{12} & \cdots & w_{1n} \\
  w_{21} & w_{22} & \cdots & w_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  w_{n1} & w_{n2} & \cdots & w_{nn}
\end{bmatrix}
\]

\[
\vec{w}_{ik} = \begin{bmatrix}
w_{1ik} \\
w_{2ik} \\
\vdots \\
w_{nik}
\end{bmatrix}
\]

\[
W' = \begin{bmatrix}
w'_{11} & w'_{12} & \cdots & w'_{1n} \\
w'_{21} & w'_{22} & \cdots & w'_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
w'_{n1} & w'_{n2} & \cdots & w'_{nn}
\end{bmatrix}
\]

\[
\vec{w}'_{ik} = \begin{bmatrix}
w'_{1ik} \\
w'_{2ik} \\
\vdots \\
w'_{nik}
\end{bmatrix}
\]

(98)

Represent the vector \( \vec{\xi}^{(k)} \) in expanded form:

\[
\vec{\xi}^{(k)} = \begin{bmatrix}
\xi_1^{(k)} \\
\xi_2^{(k)} \\
\vdots \\
\xi_n^{(k)}
\end{bmatrix}, \\
\left\{ \begin{array}{l}
\delta_1^{(k)} = (w_{11k}, w_{12k}, \ldots, w_{1nk}) \\
\delta_2^{(k)} = (w_{21k}, w_{22k}, \ldots, w_{2nk}) \\
\vdots \\
\delta_n^{(k)} = (w_{n1k}, w_{n2k}, \ldots, w_{nnk})
\end{array} \right.
\]

(99)

Let us denote

\[
W \cdot \vec{\xi}^{(k)} = \vec{\xi}^{(k)}
\]  

(100)

\[
\vec{\epsilon}^{(k)} = \begin{bmatrix}
\epsilon_1^{(k)} \\
\epsilon_2^{(k)} \\
\vdots \\
\epsilon_n^{(k)}
\end{bmatrix}, \\
\left\{ \begin{array}{l}
\delta_1^{(k)} = (w'_{11k}, \vec{\xi}^{(k)}) \\
\delta_2^{(k)} = (w'_{21k}, \vec{\xi}^{(k)}) \\
\vdots \\
\delta_n^{(k)} = (w'_{n1k}, \vec{\xi}^{(k)})
\end{array} \right.
\]

(101)

\[
c^{(k)} = (\vec{\xi}^{(k)}, \vec{\epsilon}^{(k)})
\]  

(102)

\[
k^{(k)} = (\vec{\epsilon}^{(k)}, \vec{\epsilon}^{(k)})
\]  

(103)

\[
\mu^{(k)} = \frac{c^{(k)}}{k^{(k)}}
\]  

(104)
Determination of the components of the vector $\mathbf{a}^{(k)}$, vector $\mathbf{b}^{(k)}$, the quantities $c^{(k)}$ and $d^{(k)}$ is carried out by calculating the scalar product of two vectors. Therefore, it is advisable to perform all the above calculations in the group operation processor. Calculations by formulas (95), (104), (105) must be performed in the CPU.

After the computing system finishes one of the modelling stages, which involves the visualisation of three-dimensional scenes of the concepts obtained, the process of forming stereo display files begins. Computational procedures that are associated with the transformation of spatial images are an integral part of the interactive mode of graphic display systems in general and stereographic systems in particular, since they serve the purpose of better perception of complex three-dimensional pictures. The operation of transforming three-dimensional machine images, such as transformations, rotations, scaling, mirroring, etc., is usually carried out in the so-called uniform coordinates, where a point in three-dimensional space is represented by a four-dimensional string vector $\begin{pmatrix} X \ Y \ Z \ H \end{pmatrix}$ or column vector; – some arbitrary number. It is shown that the transformation operations of three-dimensional machine images can be performed on the basis of the generalised transformation matrix $T$ of size (4x4):

$$
T = \begin{bmatrix}
a & b & c & p \\
d & e & f & q \\
h & i & j & r \\
l & m & n & s
\end{bmatrix}
$$

Below are some examples of particular types of the matrix $T$ for the case when a point in three-dimensional space is represented by a four-dimensional column vector with $H = 1$:

$T_1$ – partial rescaling matrix;
$T_2$ – general rescaling matrix;
$T_3$ – three-dimensional shift matrix;

$$
T_1 = \begin{bmatrix}
a & 0 & 0 & 0 \\
0 & e & 0 & 0 \\
0 & 0 & f & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
T_2 = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
T_3 = \begin{bmatrix}
1 & b & c & 0 \\
d & 1 & f & 0 \\
h & l & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

$T_4$ – matrix of three-dimensional rotation around the $OX$ axis by $\angle \theta$;
$T_5$ – matrix of three-dimensional rotation around the $OY$ axis by $\angle \varphi$;
$T_6$ – matrix of three-dimensional rotation around the $OZ$ axis by $\angle \phi$;
$T_7$ – matrix of rotation by $\angle \theta$ around an arbitrary axis of rotation given by the direction cosines $n_1 = \cos \alpha, n_2 = \cos \varphi, n_3 = \cos \phi$.
The examples considered are quite enough to make sure that a wide range of graphic transformations can be carried out on the basis of a square matrix of the fourth order; and that the implementation of the considered transformations is associated with the group operation of scalar multiplication of two real vectors.

In computational procedures related to 3D development based on the proposed operator method showed that in them the indicated group operation is also the most time-consuming computational operation. This applies to:

– procedures that implement forward and backward stereo operators of constant variable and rotary angles of stereo modelling – procedures related to the analysis of modelling zones;

– procedures for analysing geometric distortions in various stereometric systems;

– procedures for modelling high-rise buildings, interpolation, movement, video environment of the area.
4. Conclusion
The analysis of the listed computational procedures showed that the most time-consuming computational operation in them is the group arithmetic operation of scalar multiplication of two real vectors, which is dominant in many computational processes of algebra, analysis, in matrix transformations, in 3D graphics and in spatial modelling of urban environment. For those specialized stereometric systems that are included in the operational control loop of important critical dynamic processes, a specialized processor of the sums of paired applications will significantly increase the computing performance in the procedures of the proposed urban environment model, which ultimately, if other necessary above mentioned conditions are met, will lead to an increase in the speed of adequate perception of wireframe or full-scale wireframe models of vertical design of the urban environment.

References


