Signal-oriented processing in a speed independent environment

Aleksandr Katkow
University of Computer Sciences and Skills, Lodz, Poland
aleksandr.katkow@yahoo.com

Abstract: This article is devoted to the research of a simulation system that uses signal-oriented integrators. The simulation environment is based on SIC (Speed Independent Circuits). The article first examines the implementation of arithmetical operations, which have been inspired by the natural world, in environments lacking global synchronization of computational processes. The article then describes the process used to calculate the Riemann integral using the signal oriented integrator and develops integration algorithms (rectangles and trapezoids) for use in such environments. It includes pseudo-code that implements these algorithms. A method is then discussed for fixing the duration of transients in combinational logic circuits and for a signal-oriented implementation of the integration process. It also considers the mapping of the signal oriented integration process as a fuzzy integration process. Finally, the article presents the results of a computer simulation of a system with signal-oriented integrators for solving differential equations of partial derivatives.

Keywords: Signal Oriented Processing, Speed Independent Circuits, Differential Equations Solving

1. Introduction

In this paper, we consider a system of inertial elements that is configured using input signals and run on technology circuits that work independently of synchronization rate. These circuits are called Speed Independent Circuits (SIC) and were proposed by D. E. Muller in [1]. A very important feature of these systems is that their computational processes are not synchronised globally. This allows for a much higher computing speed than in conventionally synchronized structures. Naturally, this synchronization process delays the computational process' evolution. Furthermore, the synchronization interval is always selected to be larger than the computational interval of the slowest element within the computing environment. The computational process in systems with functional elements that are independent of the process' synchronization rate is performed using a rate determined by a realistic (i.e. inspired by the natural world) work interval of each functional element. Therefore, SIC refers to a class of VLSI combinational logic, which generates a signal when the transient process has completed.

The recent development of massively parallel computational systems has resulted in two general methods of use. One of these focuses on the accuracy of calculations while the second involves the implementation of extremely fast but approximate calculations. The results of this paper may be considered as an extension of the investigations of G. M. Baudet [5] applied to special-purpose multiprocessors systems that simulate dynamic processes. This paper considers a specialized multiprocessor system that uses signal-oriented integrators as inertial elements.
The functional elements of the computational system interact in a signal-oriented manner, with the operands within the memory blocks at the inputs of the functional elements being loaded after the transients within the functional elements have terminated. The duration of the transitional process of each functional element is defined by both the physical properties of its components and the type of operands at the inputs. A similar situation occurs within the asynchronous sequential circuits (i.e. asynchronous finite state machines (FSMs)), S. H. Unger [2].

The main difference between environments constructed over an SIC foundation and asynchronous sequential circuits is that for FSMs to function correctly, they must be provided with a special sequence of input signals. The requirement is that at any point in time the operand may change on only one of the inputs of the sequential circuits. It is practically possible to implement such a procedure for changing the input signal, but this can significantly reduce the rate of information processing. In computing environments based on SIC technology, such a limitation does not exist since such systems can be initially implemented with self-synchronization processes, which allow them to control the downloading of input information and to signal the end of the transition process in the logical scheme.

The development in ideas involving the construction of asynchronous sequential circuits also seems to be migrating towards the possibility of creating self-synchronizing structures (i.e. self-timed systems) [3,4]. Of course, in the practical implementation of SIC, the problem still exists of how to measure the duration of the transient process within combinational logic circuits. A description of one method to determine when the transitional process of combinational logic circuits has finished, based on the fixation of the electromagnetic radiation that is caused by changes in the states of the logical elements during the transient process, is given in [6,7]. This method is based on the physical features of the transitional process in e-logic. During execution of these logical operations, a change takes place in the states of the logic elements that leads to the generation of electromagnetic radiation from the e-logic. When the transitional processes of a logical combinational circuit terminate, the circuit stops generating electromagnetic radiation. The time of termination can be measured if it is possible to detect the electromagnetic radiation accompanying the transition process. For this purpose we can use a special electromagnetic screen that can be mounted onto the device's signal-oriented functional elements. In the class of VLSI combinational logic, such a screen could be implemented using VLSI technology.

A device that implements the above method, by fixing the end of the transitional process within combinational logic circuits, and the scheme of its implementation was proposed in [7,8]. In [9] the process of performing arithmetic operations in an SIC environment are represented as a process inspired by the natural world. It is interesting to investigate the performance of environments integrated using signal-oriented integrators (SOI). This article explores the integration of signal-oriented processes and their application to solving various boundary value problems using a parabolic differential equation.

2. Signal-oriented integration in Speed Independent Circuit

Various implementations of the signal-oriented integration procedure in an SIC environment are possible. Some peculiar examples of these implementations are as follows. Numerical integration is often used in the digital processing of images and in other advanced algorithms. As mentioned above, computing processes in SIC environments are organized differently than in traditional computing processes, since they can contain self-organizing computational processes. Similar processes exist in neural networks, in which one of the advantages is the high information processing speed and the ability to perform
calculations without the need to run algorithms. Information processing within the neural network structure is performed asynchronously in each group of neurons. The self-synchronization process is determined by the physical properties of each neuron. When the sum of the input signals to a neuron exceeds the threshold, the neuron generates an output signal and ignores any further inputs. Only after an output signal has been generated and the result of the operation across the whole neural network has been sent, does a neuron go into a state when it can again analyze the information that is on its inputs. Similar processes occur in computing environments with SOI, where the duration of operations for computational components is random.

For example, starting with the simplest method of numerical integration, which is the rectangular method. Traditional integration using the rectangular method involves the application of equal integration intervals for each step. In computing environments with SOI, the interval of integration for each step can arbitrarily lie between \( \tau_{\text{min}} \) and \( \tau_{\text{max}} \). Suppose that \( F(X) \) is a real integrable function from \( X \subset \mathbb{R}^1 \) to \( Y \subset \mathbb{R}^1 \) and that it is necessary to calculate the integral on a closed interval \( [a,b] \in X \). We introduce \( n+1 \) real numbers such that \( (t_0,t_1,...,t_n) \in [a,b] \) and \( a = t_0 < t_1 < ... < t_{n-1} < t_n = b \). Riemann’s ordinary integral is defined by:

\[
I(a,b) = \int_a^b f(t) \, dt = \lim_{n \to \infty} S_n
\]

where

\[
S_n = \sum_{i=1}^{n} (t_i - t_{i-1}) f(\eta_i), \quad \eta_i \in [t_{i-1}, t_i]
\]

For the implementation of Riemann’s integrals applied to SOI, the duration of execution of the computational process is unknown and for this reason needs to be measured. Let \( \tau_i = t_i - t_{i-1} \). The main problem of calculating the Riemann integral for the SOI structure is as follows. We cannot multiply by the interval \( \tau_i \), since its duration is unknown. One possible solution to this problem is to multiply the sample of the function by the interval of the time executing operation, which took place in an earlier step. For this reason we measure the duration of the first step of integration and then multiply the sample of the function \( f(t_0) \) by \( M(\tau_i) \), where \( M(\tau) \) is the expected value of the interval of time that it takes to produce the multiplication samples of \( f(t_i) \) in the interval \( \tau_i, i \in \{1,...,n\} \). Thus, we perform multiplication on the sample function using the duration of the time interval of calculations produced by the previous step. This produces a formula of bottom rectangles and the multiplication executes the duration of time from the preceding step.

The sequence of processes is represented in Table 1.

<table>
<thead>
<tr>
<th>Interval</th>
<th>Multiplication</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 - t_0 )</td>
<td>( M(\tau) )</td>
<td>( \tau_1 )</td>
</tr>
<tr>
<td>( t_2 - t_1 )</td>
<td>( \tau_1 )</td>
<td>( \tau_2 )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>( t_{n-1} - t_{n-2} )</td>
<td>( \tau_{n-1} )</td>
<td>( \tau_n )</td>
</tr>
<tr>
<td>( t_n - t_{n-1} )</td>
<td>( \tau_n )</td>
<td>-</td>
</tr>
</tbody>
</table>
A characteristic point is that for every elementary interval, it is necessary to execute multiplication on the sample of a function over the interval, the value of which is changed randomly and is unknown when the sample of the function is taken. In formula (1), the inequality $\tau_{\text{min}} \leq \eta_i \leq \tau_{\text{max}}$ holds for each integration step. The integration process is performed in accordance to the formula

$$
\tilde{S}_0 = f(t_0)M(\tau), \\
\tilde{S}_1 = \tilde{S}_0 + f(t_1)\tau_1, \\
\tilde{S}_2 = \tilde{S}_1 + f(t_2)\tau_2, \\
\vdots \\
\tilde{S}_n = \tilde{S}_{n-1} + f(t_n)\tau_n,
$$

where $t_0 = t_a$, $t_n = t_b$.

Over all intervals $[a, b]$, the formula can be recorded as

$$
\tilde{S}_n = f(t_0)M(\tau) + \sum_{i=1}^{n} f(t_i)\tau_i, \quad \tau_i = t_i - t_{i-1}
$$

where $\tilde{S}_n$ is the approximate value of the sum $S_n$.

A graphical representation of the integration process is shown in Fig. 1.

![Diagram of the rectangle method across a random interval](image)

Figure 1. Diagram of the rectangle method across a random interval: a) $\tau_{i+1} < \tau_i$, b) $\tau_{i+1} > \tau_i$

An algorithm that simulates Riemann’s type integration using the rectangle method for a signal-oriented structure with a random elementary interval of integration is introduced below:

**Input:**
- $f(t)$ – function
- $T$ – interval of analysis
- $\tau_{\text{min}}, \tau_{\text{max}}$ – minimum and maximum intervals of the computational process duration in an SOI circuit
- $M(\tau)$ – expected value of computational interval

<image>
Output:

\[ F(t) \]

\[ F(T) = 0, \quad t = 0 \]

while \( t \leq T \) do

\[ r = \text{random from } \tau_{\min} \text{ to } \tau_{\max} \]

if \( t = 0 \)

\[ \text{calculate } f(r) \]

\[ \text{calculate } F(t) = f(r) \times M(\tau) \]

end if

if \( t > 0 \)

\[ \text{calculate } f(t) \]

\[ \text{calculate } F(t) = F(t) + f(t) \times r \]

\[ t = t + r \]

end if

end while

It is evident that the formulas for the method of lower rectangles in (3) contain a multiplicative operation. When integration needs to be quickly performed (for example, when processing images in-line), the calculations can be accelerated by replacing the multiplication of the signal samples at random intervals with the multiplication of the signal samples at the expected value of the elementary interval of integration \( M(\tau) \). The operation of the integration, in this instance, is an additive operation on a collection of signal samples in compliance with the formula:

\[ \bar{S}_0 = f(t_0)M(\tau), \]
\[ \bar{S}_1 = \bar{S}_0 + f(t_1)M(\tau), \]
\[ \bar{S}_2 = \bar{S}_1 + f(t_2)M(\tau), \]
\[ \ldots \]
\[ \bar{S}_n = \bar{S}_{n-1} + f(t_n)M(\tau) \]

or otherwise

\[ \bar{S}_n = M(\tau) \sum_{j=0}^{n} f(t_j) \]  \hspace{1cm} (4)

where \( M(\tau) \) is a mathematical expectation of the intervals \( \tau, t \) – i.e. a point which may be either inside or outside of the interval \([t_{i-1}, t_i]\).

An algorithm that simulates the rectangle method across a random interval is introduced below.

Input:
\[ f(t) \] – function
\[ T \] – interval of analysis
\[ \tau_{\min}, \tau_{\max} \] – minimum and maximum intervals of the computational process duration in an SOI circuit

Output:

\[ F(T) \]

\[ F(T) = 0, \quad t = 0, \quad i = 0, \quad R = 0, \quad M(\tau) = 0 \]

while \( t \leq T \) do
\[i = i + 1\]
\[r = \text{random from } \tau_{\text{min}} \text{ to } \tau_{\text{max}}\]
\[\text{calculate } f(t)\]
\[\text{calculate } F(T) = F(T) + f(t)\]
\[t = t + r\]
\[R = R + r\]
\[\text{end while}\]
\[\text{calculate } M(r) = \frac{R}{i}\]
\[\text{calculate } F(T) = F(T) \times M(r)\]

A graphical representation of the integration process is shown in Fig. 2.

![Graphical representation of integration process](image)

Figure 2. The method of rectangles with \(M(r)\) interval: a) \(M(r) > \tau_1, \tau_2, \ldots\), b) \(M(r) < \tau_1, \tau_2, \ldots\)

The implementations of the trapezoid algorithm in SOI structures are similar. As with the algorithm for rectangles, the elementary integration interval is a random value ranging between \(\tau_{\text{min}}\) and \(\tau_{\text{max}}\). The integral in (1) modified for the method of trapezoid is calculated according to the formula:

\[
S_n = \sum_{i=1}^{n} \frac{1}{2} (f(t_i) + f(t_{i+1})) \tau_i, \quad \tau_i = t_{i+1} - t_i
\]  

(5)

As was the case earlier, the first interval of the computational process is produced by multiplying the samples of the function by the expected value of the interval of duration of the computational process. For the subsequent intervals, the multiplication is performed on the duration of the interval of time from an earlier step. The process of calculating the integral is as follows:

\[
\tilde{S}_0 = \frac{1}{2} (f(t_0) + f(t_1)) M(r),
\]

\[
\tilde{S}_1 = \tilde{S}_0 + \frac{1}{2} (f(t_1) + f(t_2)) \tau_1,
\]

\[
\vdots
\]

\[
\tilde{S}_n = \tilde{S}_{n-1} + \frac{1}{2} (f(t_n) + f(t_{n+1})) \tau_n
\]
The simulation algorithm that uses the trapezoid method for SOI circuits is introduced below:

**Input:**
- \( f(t) \) – function
- \( T \) – interval of analysis
- \( \tau_{\text{min}}, \tau_{\text{max}} \) – minimum and maximum intervals of the computational process duration in an SOI circuit

**Output:**
- \( F(T) \)

\[
F(T)=0, \ t=0, \ i=0, \ \tau=0
\]

while \( t \leq T \) do
  \( \tau=\text{random from } \tau_{\text{min}} \text{ to } \tau_{\text{max}} \)
  if \( t=0 \)
    calculate \( f(t) \)
    calculate \( f(t+\tau) \)
    calculate \( F(t)=\frac{1}{2}*(f(t)+ f(t+\tau))* M(\tau) \)
  end if
  if \( t>0 \)
    calculate \( f(t) \)
    calculate \( f(t+\tau) \)
    calculate \( F(t)=F(t)+ \frac{1}{2}*(f(t)+ f(t+\tau))* \tau \)
    \( t=t+\tau \)
  end if
end while

A graphical representation of the integration process is shown in Fig. 3.

![Figure 3. The trapezoid method over random intervals](image)

A defect of the implementation of the trapezoid method when applied to SOI circuits is that the computation of the product of the elementary interval and the average of samples of the function takes significant time, slowing down the calculation. A significant acceleration at a cost of a smaller precision in the integral's calculation using the trapezoid method can be achieved by changing the random interval to an interval equal to the mathematical expectation, \( M(\tau) \).
The operation of integration in this instance is the additive operation on a large amount of signal samples, according to the formula:

\[
\ddot{S}_0 = (f(t_0) + f(t_1))M(\tau)/2,
\ddot{S}_1 = \ddot{S}_0 + (f(t_1) + f(t_2))M(\tau)/2,
\ddot{S}_2 = \ddot{S}_1 + (f(t_2) + f(t_3))M(\tau)/2,
\ldots
\ddot{S}_n = \ddot{S}_{n-1} + (f(t_n) + f(t_{n+1}))M(\tau)/2
\]

The formula (6) can also be written as:

\[
\ddot{S}_n = (f(t_0) + f(t_{n+1}))/2 + M(\tau)\sum_{j=1}^{n} f(j),
\]

where \( t_j \) is a point which may lie either inside or outside of the interval \([t_{i-1}, t_i]\).

The simulation algorithm that uses the trapezoid method with an SOI circuit is introduced below:

**Input:**
- \( f(t) \) – function
- \( T \) – interval of analysis
- \( \tau_{\min}, \tau_{\max} \) – minimum and maximum intervals of the computational process duration in an SOI circuit

**Output:**
- \( F(T) \)

\[
F(T)=0, \ t=0, \ i=0, \ R=0, \ M(\tau)=0,
\]

while \( t \leq T \) do
  if \( t=0 \)
    calculate \( f(t) \)
    \( F(T)=f(t)/2 \)
  end if
  \( r=\text{random from } \tau_{\min} \text{ to } \tau_{\max} \)
  \( t=t+r \)
  \( R=R+r \)
  calculate \( f(t)=f(t+r) \)
  calculate \( F(T)=F(T)+f(t) \)
  if \( t = T \)
    calculate \( f(t) \)
    \( F(T)=F(T)+f(t)/2 \)
  end if
  calculate \( M(\tau)=R/i \)
end while

\[
F(T)=F(T) \times M(\tau)
\]

For the research, we executed a computer simulation of the signal-oriented integration procedure. Fig.4 a,b presents an example of the computation of the antiderivative of the function \( f(x) = \sin x \) over the interval \([0,4]\). Fig. 4c presents the distribution of the values of an antiderivative of the function \( f(x) = \sin x \) at the point \( x = 4.0 \), obtained as the result of 500 executions of signal-oriented integration by means of the rectangle method (4).
Signal-oriented processing in a speed independent environment

The distribution of the time interval of the transition process within the e-logic circuits conforms to the normal distribution. In this study we used a standard random number generator implemented in a Borland C++ environment.

3. Signal-oriented integration and fuzzy integration

The initial introduction of the concept of the fuzzy set by L. A. Zadeh in 1965 [10] has been followed by numerous theoretical and practical developments in ensuing years. The concepts of fuzzy numbers and operations over them, as introduced by S. Nahmias [12] and D. Dubois and H. Prade [11], have provided powerful tools for numerical theories and practical applications. Further developments in their applications have involved the main notions of mathematical analysis such as fuzzy integration and differentiation of fuzzy-valued mappings, which have been considered by D. Dubois and H. Prade [11]. Developments in the theory of fuzzy sets have also been used in systems analysis (J. Kasprzyk [13]), in the theory of fuzzy-control (A. Piegat [14]), and in artificial intelligence (L. Rutkowski [15]).

A variant of a signal-oriented integration, which complies with formula (3), can be written in terms of the integration of fuzzy mappings in the style of D. Dubois and H. Prade. In fact, executions of signal oriented-integration using SIC do not compute the integral of the input function $f(x)$, but instead some other integral of another function, which may be considered as a fuzzy function having fuzzy properties, due to the fuzziness of its argument. As we do not know the exact value of a point, we do not know the exact value of the function $f(x)$. It is therefore more natural to consider the point not as an ordinary real number but as a fuzzy number or more precisely as a fuzzy point in the sense of [11], i.e. a point which is not precisely located but whose position is fuzzily restricted.

So the function $f(x)$ may be considered as a fuzzy mapping $\tilde{f}$ from the interval $[a,b]$ to a fuzzy subset in $\mathbb{R}$ with a membership function $\mu_{f(x)}$. For such a fuzzy function $\tilde{f}(x)$ D. Dubois and H. Prade have introduced the notion of a fuzzy integral $\tilde{I}_{(a,b)}$ as a fuzzy set $\mathbb{R}$ with a membership function:

$$\mu_{\tilde{I}_{(a,b)}}(T) = \sup_{g \in G:T = \int_a^b g(x)dx} \mu_{\tilde{f}(g)}$$

with

$$\mu_{\tilde{f}(g)} = \inf_{x \in [a,b]} \mu_{\tilde{f}(x)}(g(x)).$$
This fuzzy integral may be approximated using a fuzzy Riemann sum:

\[ \tilde{\Sigma} = \oplus \sum_{i=1}^{n} (t_i - t_{i-1}) \tilde{f}(t_i). \]

The sum of real numbers in formula (3) is modified by an extended sum of fuzzy numbers as described in [11].

For signal-oriented integration, in accordance with Formula (4) and in terms of the fuzzy number arithmetic, the value of an elementary interval of integration becomes replaced by the expected value for the elementary integration interval. This interval depends on many random factors, such as the technology used for production of integrated circuits and the type of the operands used as integrator inputs. In the simulation using the standard procedure for generating random numbers and in Fig. 5a the statistical distribution can be seen of the values of the elementary interval of integration. The uncertainty associated with the value of the elementary integration interval is described in fuzzy number terms, by entering the normalized membership function \( \mu(x) \) for a fuzzy number \( \tilde{\mu} \), and by mapping the expected duration of the elementary interval of integration \( M(x) \). The mathematical expression for the membership function can be represented as:

\[
\mu_{\tilde{\mu}}(x) = \begin{cases} 
\frac{x - (m - \alpha)}{\alpha} & \text{for } x \in [m - \alpha, m] \\
\frac{(m + \beta) - x}{\beta} & \text{for } x \in [m, m + \beta] \\
0 & \text{otherwise}
\end{cases}
\]

where \( m = M(x) \), which is the mean value of a fuzzy number \( \tilde{M} \), \( \alpha \) – left spread, \( \beta \) – right spread. Thus, the interval duration can be represented by the fuzzy number \( \tilde{\mu} \), with a membership function L-R type \( \mu_{\tilde{\mu}}(x) \) [11,13]. The normalized membership function can be represented by \( \mu_{\tilde{\mu}}(x) \) for a fuzzy number \( \tilde{\mu} \), and determined using the elementary interval of integration in the present variant. This signal-oriented integration is shown in Fig.5b. If the integrand is not converted to a fuzzy number, the calculation of the integral value for each elementary interval of the integration process can be performed by calculating the product of the sum of non-fuzzy values of the integrand on a fuzzy number \( \tilde{\mu} \), which represents an elementary interval of integration. The product of fuzzy value duration of the integration and the non-fuzzy value of the integrand is also a fuzzy number. Thus the value of the fuzzy Riemann integral is a fuzzy number, which can be written as:
Signal-oriented processing in a speed independent environment

$$\overline{S}_{\tau_i} = \left( \sum_{i=0}^{n} f(x_i) \right) \otimes \overline{M}_\tau$$

with a membership function of $\mu_{M(\tau)}(x)$, as presented above.

4. Computer simulation

The aim of our computing experiments was to study the behaviour of simulation systems with signal-oriented inertial functional elements when solving the Dirichlet problem for the Laplace equation on a rectangular domain of a plane by reducing it to a parabolic differential equation. The following equation:

$$\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} = 0$$

$$v(x,y)|_G = g(x,y)$$

(7)

can be mapped onto the corresponding boundary problem for the parabolic differential equation:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

$$u(x,y,t)|_G = g(x,y)$$

$$u(x,y,0)|_G = g_0(x,y)$$

where $g(x,y)|_G = g_0(x,y)$ and $g(x,y)$ is a known function, given on the boundary $G$. The finite-difference method is used in order to obtain a numerical solution of this boundary value problem. With this goal in mind, the domain $(a \times a) \in D \cup G \subset R^2$ is covered by a mesh with step $h = a/N - 1$. The values of the functions $u(x_i,y_j), g(x_i,y_j)$ at mesh coordinates $x_i = ih, y_j = jh$ ($i,j = 0,1,...,N-1$), can be denoted by $u_{ij}, g_{ij}$. The time derivative then needs to be replaced by the difference formula, in accordance with the Euler formula. The formula of signal-oriented integration using the method of lower rectangles is used(4). For the elementary interval, this formula can be written as:

$$\frac{\partial u}{\partial t} \approx \frac{u_y(t_{k+1}) - u_y(t_k)}{M(\tau)}$$

where $M(\tau)$ is mathematical expectation of the duration of an elementary interval of the signal-oriented integration.

Partial derivatives with respect to variables $x, y$ can be written using the second order Euler difference formula:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{y-1j}(t_k) - 2u_y(t_k) + u_{y+1j}(t_k)}{2h}$$

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{u_{x-1j}(t_k) - 2u_{x}(t_k) + u_{x+1j}(t_k)}{2h}.$$
For the inner mesh points \((i, j)\), the problem (7) is replaced by the finite-difference equation.

\[
\begin{align*}
    u_y(t_{k+1}) &= u_y(t_k) + \frac{M(t)}{2h}.B(t_k), \\
    B(t_k) &= u_y(t_{k-1}) + u_{y+1}(t_k) + u_{y-1}(t_k) + u_{y+1}(t_{k+1}) - 4u_y(t_k).
\end{align*}
\]

For boundary mesh points we assume that \(u_y(x,y) = g_y(x,y)\).

The results of computer simulation for the function \(g(x,y)\) can be represented in the following form:

\[
g(x,y) = \begin{cases} 
    300 \sin \frac{\pi i}{N-1} & \text{for } x = N-1 \\
    500 & \text{if } x = 1, y = 1 \\
    500 & \text{if } x = 0, y = 15 \\
    0 & \text{otherwise}
\end{cases}
\]

In such a way a standard integration in formula (7) is replaced by signal-oriented integration. The computational process constitutes a sequence of elementary operations of signal-oriented integration that converge to a solution.

Experiments were run on systems consisting of signal-oriented inertial elements with dimensions of 16x16. Fig. 6 shows the solution to (7) obtained by performing 15 macro time steps, provided that \(M(t) = 0.5h, a = 15, N = 16\) and consequently \(h = 1\). A macro time step is defined as the time interval in which the elementary integration with respect to time process will be performed in each inertial element at least once. This means that inertial components exist within the elementary integration process that are repeatedly executed within one macro step of time. In this experiment, the end computational process occurs under the condition that the maximum value of change of the unknown function in the lattice sites of the macro step, from one macro time step to another, is no greater than \(\varepsilon_m = 0.05\). In other words:

\[
    \max_{i,j=0,...,N-1} (u_y(t_{k+1}) - u_y(t_k)) \leq \varepsilon_m,
\]

where \(k\) is the number of the macro step.
5. Conclusion

The development of SIS technology in the field of information processing has been hampered by the lack of an efficient method for determining the duration of transients in electronic logic circuits. In [6,7] a method is proposed for determining the end of the transitional process, alongside its structural implementation. [8] presents some possible solutions to this problem.

Although the working capacity of the fixation of the electromagnetic radiation accompanying the process of electronic logic gates has been experimentally verified for devices with discrete logic elements, its implementation by means of modern technologies of integrated components still remains unrealized. It is possible that with the development of new technologies it will be possible in the future to fully realize the idea of SIC. It is worth noting that in computer systems that have so far evolved, there is no global synchronization process for information processing. What we have instead are self-synchronization processes, based on the physical properties of the information processing elements.

References