NUMERICAL INTEGRATION BY GENETIC ALGORITHMS

Vladimir Morozenko, Irina Pleshkova

***Abstract**: It is shown that genetic algorithms can be used successfully in problems of definite integral calculation especially when an integrand has a primitive which can't be expressed analytically through elementary functions. A testing of the program, which uses the genetic algorithm developed by authors, showed that the best results are reached if the size of population makes 30-50 chromosomes, approximately 40-60% of its take a part in crossover, and the program stops if the population's leader didn't change during 5-10 generations. An answer of genetic algorithm is more exact than answer received by the classical numerical methods, even if a* quantity *of partition's points into segment is small or if an integrand is quickly oscillating. So genetic algorithms can compete both on the accuracy of calculations and on operating time with well-known classical numerical methods such as midpoint approximation, top-left corner approximation, top-right corner approximation, trapezoidal rule, Simpson's rule.*

***Keywords**: definite integral, integral sum, numerical integration, genetic algorithm, fitness-function.*

***ACM Classification Keywords**: F.1.2 COMPUTATION BY ABSTRACT DEVICES: Models of computation – Probabilistic computation. G.1.6 NUMERICAL ANALYSIS: Optimization – Stochastic programming.*

***Conference topic:** Genetic Algorithms and Evolutionary Computing.*

Introduction

A solution of many problems in physics, chemistry, mechanics and other natural sciences requires calculation definite integrals. Unfortunately, an exact analytic calculation of definite integrals is often impossible. Many functions do not have primitive that can be expressed analytically through elementary functions. For example, this is true for the function $f(x) = \exp(-x^2)$. Moreover, a symbolic integration is a much more difficult problem than a finding the primitive and there is no universal algorithm solving this problem. That's why an exact calculation of definite integrals by the fundamental theorem of calculus is often difficult or impossible at all.

Traditionally, many algorithms for calculating of the integral's value are used in numerical analysis. In most of them, the integrand $f(x)$ is replaced with the approximating function $\varphi(x)$ which is easier to integrate [Samarsky, 1989].

In this paper a new method of numerical integration is described. This method doesn't require knowledge of integrand's primitive because it is based on a genetic algorithm.

Such an unusual method of numerical integration expands area of applicability of genetic algorithms, which are traditionally used for solving of optimization problems [Gladkov, 2009].

Theoretical premises

Let a function $f(x)$ be defined on segment $[a,b]$. It is required to calculate a definite integral *I* of the function $f(x)$ over the segment $[a,b]$:

$$I = \int_a^b f(x)dx. \qquad (1)$$

Let $P = \{x_0, x_1, x_2, ..., x_k\}$ be a partition of the segment $[a,b]$. It is assumed that $a = x_0 < x_1 < ... < x_k = b$. The

nodes $x_0, x_1, x_2, ..., x_k$ of partition P subdivide the segment $[a,b]$ into k small segments $[x_0, x_1], [x_1, x_2], ..., [x_{k-1}, x_k]$ so that $x_i - x_{i-1} = (b-a)/k$ for all $i = \overline{1,k}$.

According to the additive property of definite integrals we have the equality

$$\int_a^b f(x)dx = \sum_{i=1}^{k} \int_{x_{i-1}}^{x_i} f(x)dx.$$

The theorem of mean value integration states that into segment $[x_{i-1}, x_i]$ there exists a point ξ_i for which the following equality is true [Shipachev, 2005]:

$$\int_{x_{i-1}}^{x_i} f(x)dx = f(\xi_i) \cdot (x_i - x_{i-1}),\ x_{i-1} \le \xi_i \le x_i.$$

Let us pick some point c_i into each segment $[x_{i-1}, x_i]$ and then define the following integral sum $S(c_1, c_2, ..., c_k)$:

$$S(c_1, c_2, ..., c_k) = \sum_{i=1}^{k} f(c_i) \cdot (x_i - x_{i-1}). \quad (2)$$

Geometric interpretation of this integral sum (2) is shown on Fig. 1.

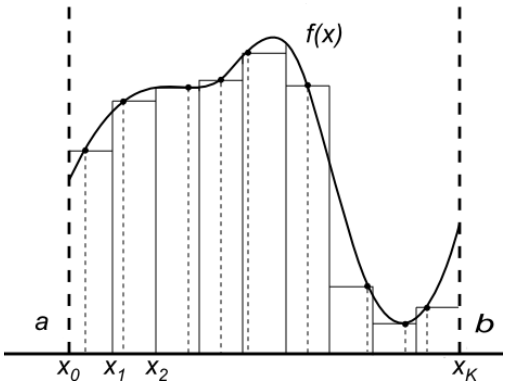

Fig. 1. Geometric interpretation of the integral sum S with random points c_i

The integral sum $S(c_1, c_2, ..., c_k)$ is random variable, because points c_i into each segment $[x_{i-1}, x_i]$ are selected by random way. However it approximately equals to the value of the definite integral (1) and the calculation error does not exceed the error given by the rectangle method. Moreover the mathematical expectation $M[S]$ of integral sum's value must be equal to the exact value of the integral (1), i.e.

$$M[S] = \int_a^b f(x)dx.$$

If another set of points c_i is selected, the value of the integral sum is different, but still it must be approximately equal to the integral's value. In case of large N arithmetic average value of integral sum S^* converges to the value of integral (1) when N tends to infinity:

$$\lim_{N\to\infty} S^* = \lim_{N\to\infty} (S_1 + S_2 + ... + S_N)/N = \int_a^b f(x)dx. \quad (3)$$

The closer value of the integral sum (2) to the average sum S^* (3) is, the closer it is to the exact value of the integral (1). Therefore if we select points fortunately, we can calculate integral (1) with high accuracy. So we

reduced the problem of the integral's calculation to the optimization problem: to find the points $c_1, c_2, ..., c_k$ into segments $[x_0, x_1], [x_1, x_2], ..., [x_{k-1}, x_k]$ so that

$$\left| S(c_1, c_2, ..., c_k) - S^* \right| \to min.$$

Further we shall show that this minimization problem can be solved by a genetic algorithm with a high accuracy. For this purpose we

- worked out a genetic algorithm for calculating the numerical value of definite integral of arbitrary functions defined on an segment;
- developed a program that implements the genetic algorithm;
- tested and debugged the program, using our genetic algorithm.

Genetic algorithm's description

Before we develop the genetic algorithm we need to:

- define the space of search;
- select the method of coding the possible solutions;
- set rules of crossover and mutation;
- define the fitness-function.

The main requirement to the fitness-function is following: its minimum's point must be an exact solution of the minimization problem [Gladkov, 2009].

By means of our genetic algorithm we will find the set of points $c_1, c_2, ..., c_k$, which minimize the difference between the integral sum (2) and the integral value (1) as far as it is possible.

Coding of possible solutions

The solution is an ordered set of *k* points, so it can be encoded by a sequence of *k* numbers – the coordinates of these points (Fig. 2). It's important to note, that there is only one point c_i into segment $[x_{i-1}, x_i]$ for each $i = \overline{1,k}$.

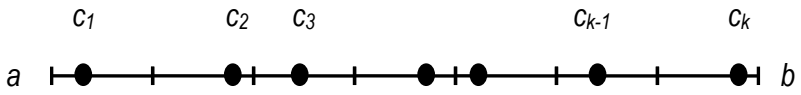

Fig. 2. The encoding of the solution

So each chromosome is a sequence of *k* numbers $c_1, c_2, ..., c_k$, where *k* is quantity of small segments into the segment $[a,b]$.

Crossover and mutation

We choose the classical crossover technique – the one-point crossover. The crossover point is a random point from the partition $P = \{x_0, x_1, x_2, ..., x_k\}$ of the segment $[a,b]$, where $x_i - x_{i-1} = (b-a)/k$ for all $i = \overline{1,k}$.

Since all the segments $[x_0, x_1], [x_1, x_2], ..., [x_{k-1}, x_k]$ have the same length, all children after the one-point crossover are viable undoubtedly, because it is impossible that there would be no point or would be more than one point into segment $[x_{i-1}, x_i]$. An example of crossover's result is shown on Fig. 3 (the vertical line cuts chromosomes).

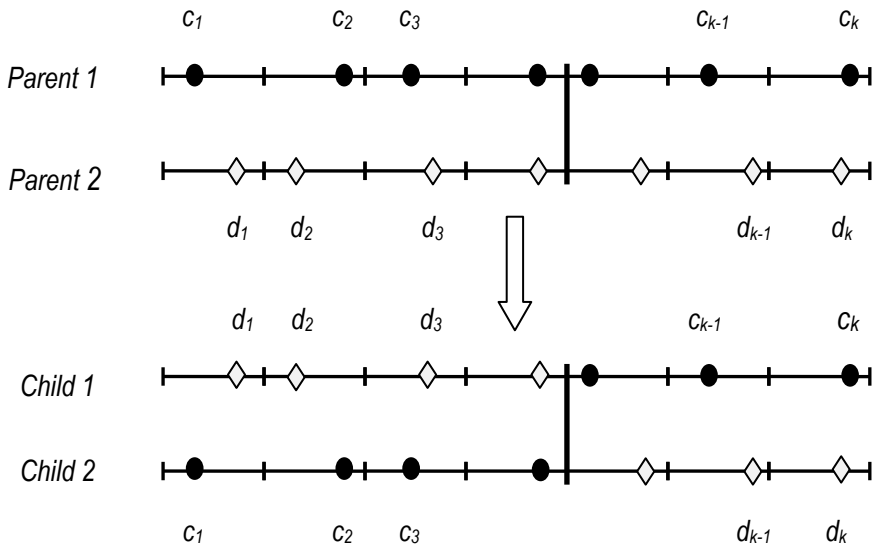

Fig. 3. One-point crossover's result

Let p_{cross} be a quantity of percents of the best chromosomes in the generation, which participate in crossover (the parameter p_{cross} can be adjusted). The crossover's point is random.

Let p_{mut} be a quantity of percents of the chromosomes, which mutate (the parameter p_{mut} can be adjusted, usually it does not exceed to 5%). The mutation's operator randomly changes the coordinate of the point c_i into segment $[x_{i-1}, x_i]$. An example of the mutation is shown on Fig. 4 (point c_3 is replaced by point d_3).

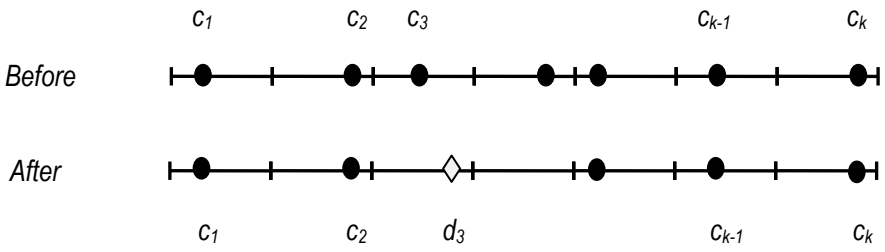

Fig. 4. Mutation's result

After the mutation operator is applied to a chromosome, the resulting chromosome always is viable, because the point c_i moves within the segment $[x_{i-1}, x_i]$.

Fitness-function

Fitness-function has to meet the following requirements: the exact decision has to settle down in a point of its global minimum and value of function has to reflect a level of fitness of a chromosome [Gladkov, 2009].

For definition of fitness-function we will calculate the integrated sum for each chromosome in population on a formula (2). The problem of optimization is to minimize a divergence between the integrated sum and the value of integral (1). As it was noted earlier, if the value of the integrated sum S is close to average value S^* of all integrated sums, then it is closer to value of integral too.

Therefore for calculation of fitness-function *F* we will use a formula:

$$F(c_1,c_2,...,c_k) = \left|S(c_1,c_2,...,c_k) - S^*\right|, \quad (4)$$

where *S** is the average value of all integrated sums in a given generation. It is obvious that the most adapted chromosomes are those who have the smallest value of fitness-function (4).

The algorithm stops when the leading chromosome doesn't change during several generations.

Testing of genetic algorithm

During a testing the quality and the operating time of genetic algorithm were investigated, their dependence on number of points in partition of a segment and parameters of genetic algorithm (such as size of the population, percent of the chromosomes chosen for crossing, a stop condition, etc.) was estimated.

To estimate the quality of the received decision in case of $I \neq 0$ the relative error of result is calculated by formula:

$$\delta = \left|\frac{I^* - I}{I}\right| \cdot 100\%, \quad (5)$$

where *I* is exact value of integral (1), *I** is answer received by means of genetic algorithm. If the integral (1) can't be calculated precisely, then *I* is received by means of some classical numerical method (such as midpoint approximation, Simpson's rule etc.). During the testing only one of parameters of the genetic algorithm was changed, and all the others parameters remained fixed.

Dependence of quality and operating time of genetic algorithm on partition's size

It is obvious that if number *k* of points into segment [*a*,*b*] grows, the value $\Delta = (b-a)/k$ decreases. Therefore it is possible to calculate the integral (1) more precisely if we increase the number *k*. Suitable value of *k* depends on properties of integrand $f(x)$.

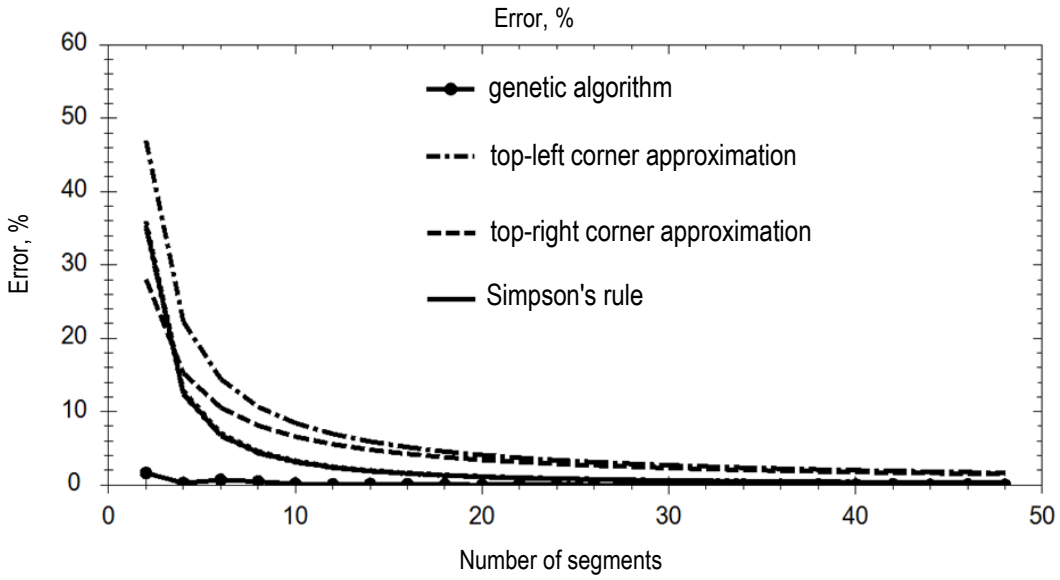

Fig. 5. A result's precision given by genetic algorithm and by numerical methods

For example we consider an integral with a monotonic continuous integrand such as $\int_0^1 \sqrt{x}dx = 2/3$. The dependence of result's precision (5) given by genetic algorithm and by numerical methods on quantity of partition's points is presented on Fig. 5. It is important to note that the result's precision (5) given by genetic algorithm is much higher than ones given by known numerical methods such as top-left corner approximation, top-right corner approximation, Simpson's rule, especially in case of small partition's points.

The essential error arises in case of quickly oscillating functions if the integral is calculated by means of classical numerical methods. For example, we mean the integral $\int_{0.01}^1 sin(1/x)dx$. The diagram of quickly oscillating integrand $sin(1/x)$ is shown on Fig. 6.

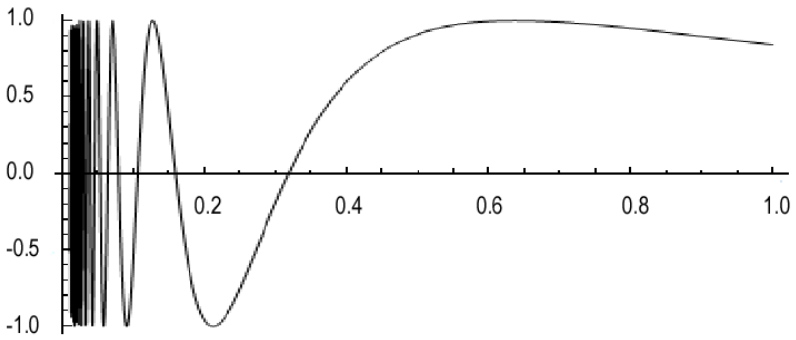

Fig. 6. Diagram of quickly oscillating function sin(1/x)

But even in case of quickly oscillating function the genetic algorithm gives more exact answer, than other known numerical methods such as midpoint approximation, top-left corner approximation, top-right corner approximation, trapezoidal rule, Simpson's rule (Fig. 7).

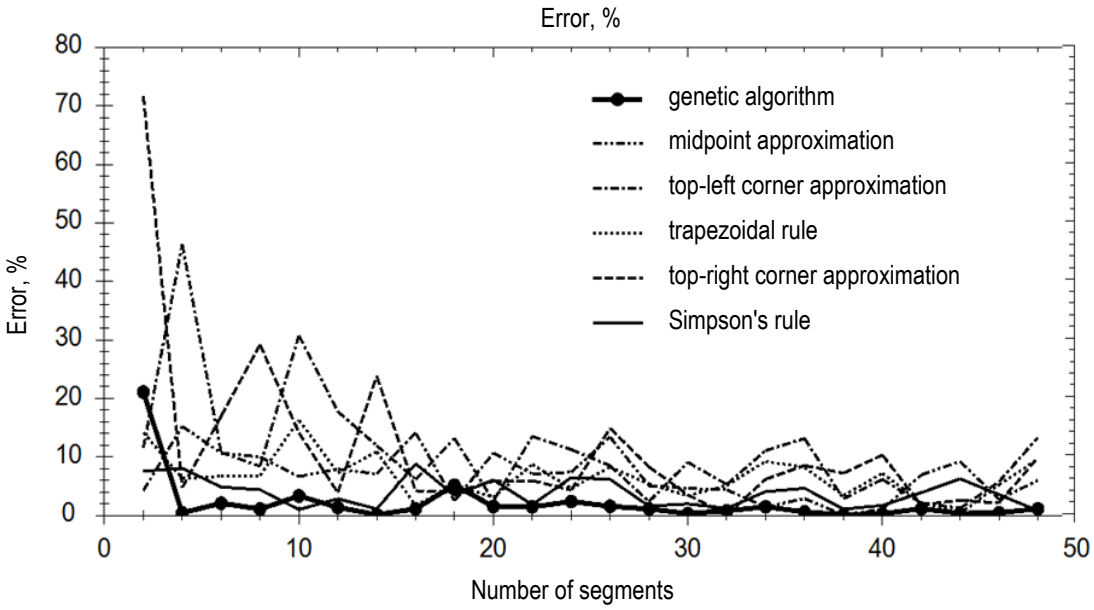

Fig. 7. Precision given by genetic algorithm and by numerical methods in case of a quickly oscillating integrand

If the quantity *k* of partition's segments increases the operating time of the genetic algorithm increases too of course. For example, if the number of partition's segments is equal to 1000 the operating time of the genetic algorithm doesn't exceed 1 minute. But if the number of partition's segments is less than 100, an operating time doesn't exceed 4 seconds.

Dependence of a quality and an operating time of genetic algorithm on its parameters

As a result of testing of the program using the genetic algorithm the following regularities were found:

- In case of small quantity of chromosomes in population (less than 15) the error of the decision is rather great. If to increase number of chromosomes in population then the average error (5) monotonically decreases and stabilizes.
- The percent of the chromosomes who are taking a part in crossover slightly influences an operating time of algorithm and doesn't influence almost response accuracy.
- The algorithm stops if the leader doesn't change during several generations. Such quantity of generations is one of the algorithm's parameter. A user can set its value voluntarily. If this parameter's value is small (less than 5), then an error of received decision (5) can be big. However in case of this parameter's value is more than 10 an error considerably decreases, and in case of its further increase the error practically doesn't change.

Calculating multiple integrals

The developed genetic algorithm can be extended for calculating multiple integrals. The task is to compute an integral where integrand depends on *n* arguments and exists into a domain *D*, where $D \subset R^n$:

$$\int_D f(x_1, x_2, ..., x_n) dx_1 dx_2 ... dx_n, \quad (6)$$

The domain of the function $f(x_1, x_2, ..., x_n)$ is a set *D* which satisfies the following requirements:

- *D* is bounded in R^n, i.e. $\exists I^n : D \subseteq I^n$, where I^n – *n*-dimensional parallelepiped;
- the bound of *D* is a null-set in Lebesgue measure.

At the beginning we consider a simple case when the domain *D* is a *n*-dimensional parallelepiped. For clarity, we describe a genetic algorithm when integrand has two arguments and *D* is a rectangle.

Integration over a rectangle

The domain *D* of the function *f*(*x*,*y*) is a rectangle which is separated into equal rectangles by lines $x = x_i, i = \overline{0, k_x}, y = y_j, j = \overline{0, k_y}$ Then we pick a point c_{ij} in each rectangle (Fig. 8).

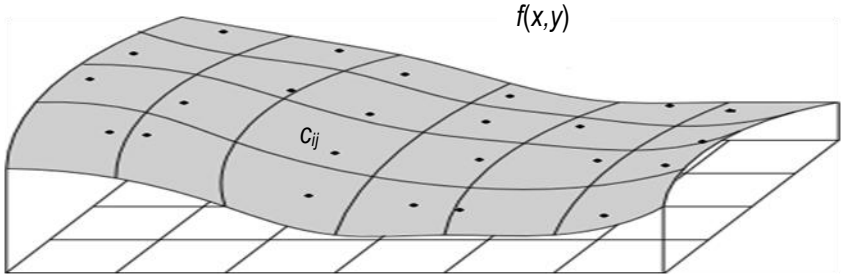

Fig. 8. Diagram of function f(x,y), the partition of the rectangle and chosen points c_{ij}

In case of $n = 2$ a chromosome is a two-dimensional array and selected points $\left\{c_{ij} = (x_i, y_j) \,\middle|\, i = \overline{0,k},\ j = \overline{0,m}\right\}$ are elements of this array. In this case we need to select 2 crossover lines for the crossover operator. But when D is n-dimensional parallelepiped we need to select n hyperplanes. Children inherit fragments of parents' chromosomes. The example of crossover is shown on Fig. 9.

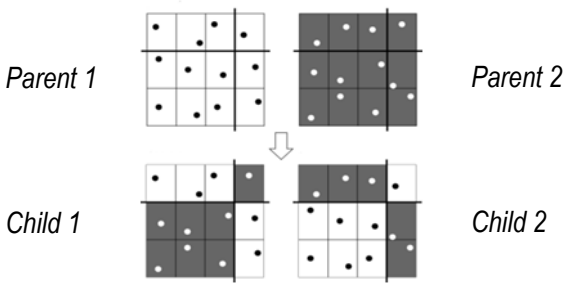

Fig. 9. Crossover operator

During the mutation a point randomly changes its position inside the rectangle (Fig. 10).

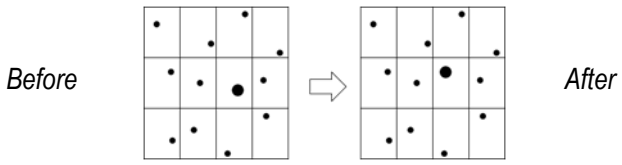

Fig. 10. Mutation operator

Fitness-function F is computed by the formula (4), where S^* is the average value of the integral sums in the generation and S is integral sum for the chromosome calculated with following formula:

$$S = \sum_{i=1}^{k}\sum_{j=1}^{m} f(c_{ij}) \cdot (x_i - x_{i-1}) \cdot (y_j - y_{j-1}) \cdot$$

Integration over an arbitrary domain

For calculating integrals (6) over more complicated areas D, we introduce the concept of characteristic function. Let *characteristic function* for the set D be the function

$$\chi_D(x) = \begin{cases} 1, & x \in D, \\ 0, & x \notin D. \end{cases}.$$

The integral (6) of the function $f(x)$ over the domain D is defined as:

$$\int_D f(x_1, x_2, \ldots, x_n) dx_1 dx_2 \ldots dx_n = \int_{I^n} f(x_1, x_2, \ldots, x_n) \cdot \chi_D(x_1, x_2, \ldots, x_n) dx_1 dx_2 \ldots dx_n \cdot$$

where $D \subseteq I^n$ and I^n – n-dimensional parallelepiped.

Thereby, the task of integration over an arbitrary domain D is reduced to the integration over a parallelepiped I^n.

We tested our genetic algorithm with the purpose to evaluate its accuracy and an operating time. The testing procedure is similar to the one-dimensional case above. One of the parameters is changing while the others are fixed.

First, we investigated the dependence of the solutions' precision on number of the segments in the partitions. These solutions were obtained by genetic algorithm and classical numerical methods. There are two examples of the test below. In the first test, the integrand is smooth continuous function which is defined on a rectangle:

$$\int_0^1 dy \int_0^1 (sin(3x) + sin(5y))dx = \frac{2}{3} sin\left(\frac{3}{2}\right)^2 + \frac{2}{5} sin\left(\frac{5}{2}\right)^2 \approx 0.8066$$

The genetic algorithm shows more accurate result than other methods, even if the number of segments is small. The dependence of the solutions' precision on number of the segments in the partitions is presented on Fig. 11.

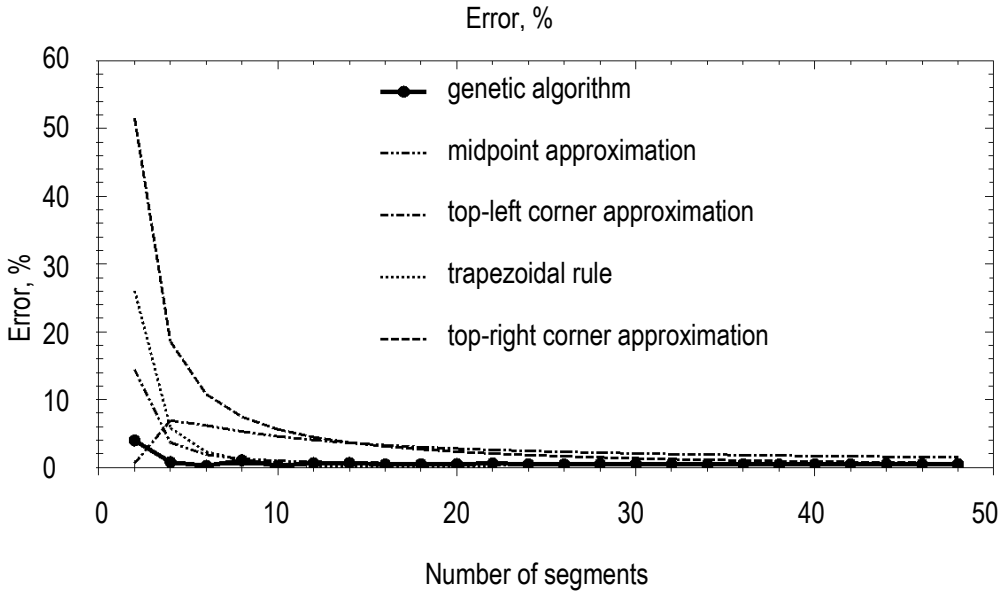

Fig. 11. Results given by the genetic algorithm and numerical methods

The second example shows us a dependence of precision on the number of segments for quickly oscillating integrand.

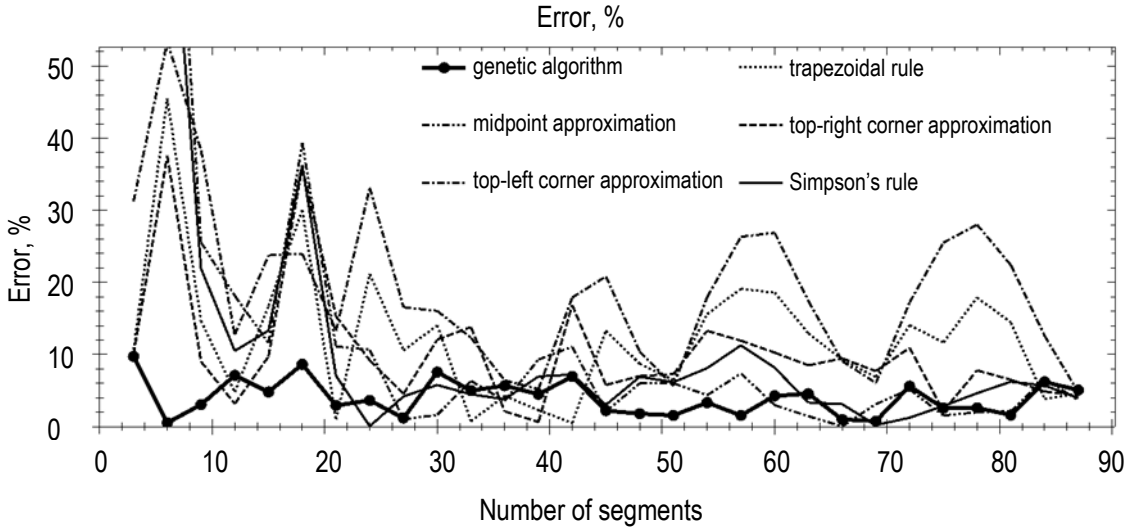

Fig. 12. Results given by the genetic algorithm and numerical methods in case of quickly oscillating integrand

In this example we calculated the integral with quickly oscillating integrand

$$\int_{0.1}^{1} dx \int_{0.1}^{\sqrt{1-x^4}} \sin\left(\frac{1}{xy}\right) dy \approx 0.023\cdot$$

The error increased when a quickly oscillating function is integrated, but the result given by genetic algorithm is more precise even than one given by Simpson's rule, not to mention other more rough approximations (Fig. 12).

Some more examples in the table below are shown:

Integral	Accuracy, %	Operating time, sec.	Quantity of generations
$\int_{\frac{3\pi}{2}}^{2\pi} d\varphi \int_{-\frac{\pi}{2}}^{0} \cos\psi d\psi \int_{0}^{2} r^2(2r\cos\varphi\cos\psi + 1)dr$	0,0088	129,64	142
$\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} d\varphi \int_{0}^{2\cos\varphi} r\sqrt{1+\cos(\varphi+\sin\varphi)}dr$	0,0783	19,34	180
$\int_{0}^{2\pi} d\varphi \int_{0}^{1} rdr \int_{2r^2}^{\sqrt{4-r^2}} zdz$	0,1125	130,53	123
$\int_{-6}^{2} dx \int_{\frac{x}{4}-1}^{2-x} \frac{3-\sin(6x+2y)}{\ln(x+10)} dy$	0,3369	12,02	109
$\int_{0}^{2} 5dx \int_{0}^{\sqrt{4-x^2}} y\,dy \int_{0}^{\frac{x^2+y^2}{4}} dz$	0,4125	229,67	145
$\int_{0}^{4} dx \int_{\frac{3}{4}x}^{\sqrt{25-x^2}} (x+\sin y)\,dy$	0,5254	4,08	36
$\int_{0}^{2\pi} d\varphi \int_{0.00001+\sqrt{11}}^{\sqrt{12}} r\sqrt{1+\frac{r^2}{r^2-11}} dr$	1,2788	8,29	81

Conclusion

A purpose of our investigation was to research a possibility of genetic algorithms' application to a task of definite integral's computation. To do this we developed a genetic algorithm and created the software product using this algorithm.

The developed genetic algorithm allows calculating definite integrals with an acceptable accuracy. Testing of a software product showed that the best accuracy of the decision is reached if the size of population makes from 30 to 50 chromosomes, 40-60% of chromosomes participate in crossover and the algorithm stops if the leader of population doesn't change during 5-10 generations.

The genetic algorithm with the specified parameters provides the more exact result than if we would apply other well-known numerical methods such as midpoint approximation, top-left corner approximation, top-right corner approximation, trapezoidal rule, Simpson's rule. Advantage of the genetic algorithm is especially noticeable, when a quantity of partition's points is small and also when an integrand is quickly oscillating.

Executed research shows that genetic algorithms can be used for numerical integration when integrand has a primitive which can't be expressed analytically through elementary functions. Also developed genetic algorithms

allow to calculate multiple integrals with integrand function of *n* arguments defined over *n*-dimensional parallelepiped or arbitrary bounded domain into *n*-dimensional space. Thus was confirmed that genetic algorithms can successfully compete with classical numerical methods both on the accuracy of computation and on an operating time.

Acknowledgement

The paper is published with partial support by the project ITHEA XXI of the ITHEA ISS (www.ithea.org) and the ADUIS (www.aduis.com.ua).

Bibliography

[Самарский, 1989] Самарский А.А., Гулин А.В. Численные методы. – М: Наука, 1989.

[Гладков и др., 2009] Гладков Л.А., Курейчик В.В., Курейчик В.М., Сороколетов П.В. Биоинспирированные методы в отпимизации. – М: ФИЗМАТЛИТ, 2009.

[Шипачев, 2005] Шипачев В.С. Высшая математика / Учебник для вузов. – М: Высш.шк. 2005.

Authors' Information

Vladimir Morozenko *– National Research University Higher School of Economics, Department of Information Technologies in Business, Associate Professor, Perm, Russia, 614070, Studencheskaya,38; e-mail: v.morozenko@mail.ru.*

Major Fields of Scientific Research: Genetic algorithms, Combinatorial algorithms, Optimization tasks, Cryptology, Cryptography

Irina Pleshkova *– Perm State National Research University, Department of Software and Computing Systems Mathematical Support, University undergraduate, Perm, Russia, 614990, Bukireva, 15; e-mail: elf_irina@mail.ru.*

Major Fields of Scientific Research: Genetic algorithms, Optimization tasks