УДК 004.051

S.S. BRUKHANKOV[1], B.M. KONOREV[2], M.S. L′VOV[3], G.N. ZHOLTKEVYCH[4]

[1]Mega Business Software Ltd, Ukraine
[2]M. Zhukovski National Air-Space University, Ukraine
[3]Kherson State University, Ukraine
[4]V. Karazin Kharkiv National University, Ukraine

ABOUT STATIC ANALYSIS OF VARIABLES PHYSICAL DIMENSIONS FOR CRITICAL-MISSION SOFTWARE

Some strong method of static code analysis, namely, analysis of physical dimensions of program variables is researched in the paper. The formal model to analyze the dimensions of variables is built. Dimensions and dimensional distributions are presented in the paper in order to define invariants for program variables. The method to analyze the model is described and grounded in the paper. The experimental results shown in the paper make it possible to evaluate test-sensitivity at the rate of about 65%.

Key words: *software dependability, static code analysis, dimensional analysis, control flow, control flow equation, fixed point problem*

Introduction

We say that a software system is a critical-mission software system if a disastrous effect is a possible result of its running failure. Software for flight-control systems, nuclear reactor control systems, power supply control systems, medical equipment control systems and similar systems afford examples of critical-mission software systems.

We use the term "dependability" to designate those software system properties that allow us to rely on a system functioning as required [1]. Some of these properties allow checking by methods of static code analysis. Reasons for using static code analysis for critical-mission software verification are discussed in [2,3].

Authors and their partners expect that processing results of running a set of static code analysis algorithms provide assessment of software dependability [4,5].

We can define two classes of static code analysis methods, namely: weak methods and strong methods. Principal characteristic of a weak method is using programs models which are focused on computing aspects of a program. In contrast to this strong methods are based on programs models which take into account facts and relations of project scope.

Some strong method is considered in the paper. The method deals with dynamical properties of physical dimensions of variables. Namely, physical dimensions of variables should be invariant for semantically correct program [4].

Authors use the monograph of American physicist, Nobel Winner P.W. Bridgman "Dimensional Analysis" [6] as physical background of the paper.

Our aim is to build and to ground the method for checking invariance of physical dimensions of variables.

1. Programs Models

1.1. Variables

Suppose $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ are finite disjoint sets. Elements of the sets are called input, output and local variables respectively.

By definition, put

$$\mathbf{V} = \mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}. \qquad (1)$$

1.2. Basic Statements

Suggestion 1. Any permitted statement is an assignment statement

$$v := f(v_1, \ldots, v_s), \qquad (2)$$

where $v \in \mathbf{Y} \cup \mathbf{Z}$, and $\{v_1, \ldots, v_s\} \subset \mathbf{X} \cup \mathbf{Z}$, and f is a polynomial over a set of basic operations.

Suggestion 2. A set of basic operations contains the following operations and only them:

a) binary operations addition ($v := v_1 + v_2$), subtraction ($v := v_1 - v_2$), multiplication ($v := v_1 * v_2$) and division ($v := v_1 / v_2$);

b) a countable set of unary exponential operations ($v := \mathrm{pow}_n(v_1)$);

c) a countable set of unary rooting operations – $v := \mathrm{root}_n(v_1)$.

The set of polynomials is defined by recursion:

1) a variable is polynomial;

2) if $f(v_1,\ldots,v_s)$ and $g(v_{s+1},\ldots,v_{s+t})$ are polynomials, and $\{v_1,\ldots,v_s,v_{s+1},\ldots,v_{s+t}\} \subset \mathbf{X} \cup \mathbf{Z}$ then

$$f(v_1,\ldots,v_s)+g(v_{s+1},\ldots,v_{s+t}), \quad (3)$$

$$f(v_1,\ldots,v_s)-g(v_{s+1},\ldots,v_{s+t}), \quad (4)$$

$$f(v_1,\ldots,v_s)*g(v_{s+1},\ldots,v_{s+t}), \quad (5)$$

$$f(v_1,\ldots,v_s)/g(v_{s+1},\ldots,v_{s+t}) \quad (6)$$

are polynomials;

3) if $f(v_1,\ldots,v_s)$ is polynomial, $\{v_1,\ldots,v_s\} \subset \mathbf{X} \cup \mathbf{Z}$ then

$$\mathrm{pow}_n(f(v_1,\ldots,v_s)), \quad (7)$$

$$\mathrm{root}_n(f(v_1,\ldots,v_s)) \quad (8)$$

are polynomials.

Suggestion 3. Any basic assertion is a comparison:

1) $f(v_1,\ldots,v_s)=g(v_{s+1},\ldots,v_{s+t})$;

2) $f(v_1,\ldots,v_s)>g(v_{s+1},\ldots,v_{s+t})$.

Suggestion 4. A set of assertions contains the following assertions and only them:

1) false ($\bot$) and truth ($\top$);

2) if φ is an assertion then $(\neg\varphi)$ is an assertion;

3) if φ and ψ are assertions then $(\varphi\wedge\psi)$ and $(\varphi\vee\psi)$ are assertions.

1.3. Transition Systems and Control Flows

Let $\mathbf{L}$ be a finite set with two distinguished elements $\mathrm{entry} \neq \mathrm{exit} \in \mathbf{L}$, and $\mathbf{T}$ be a finite set. Elements of $\mathbf{L}$ and $\mathbf{T}$ are called control points and transitions respectively.

Let src and snk be maps from $\mathbf{T}$ to $\mathbf{L}$. Suppose $w=\tau_1\ldots\tau_k \in \mathbf{T}^+$, w is control flow from $\mathrm{src}(\tau_1)$ to $\mathrm{snk}(\tau_k)$ if for each $i=1,\ldots,k-1$ the following condition holds

$$\mathrm{snk}(\tau_i)=\mathrm{src}(\tau_{i+1}). \quad (9)$$

A system $\langle \mathbf{L},\mathbf{T},\mathrm{src},\mathrm{snk}:\mathbf{T}\to\mathbf{L}\rangle$ is a transition system if the following conditions hold

1) $\mathrm{snk}^{-1}(\mathrm{entry})=\varnothing$, $\mathrm{src}^{-1}(\mathrm{exit})=\varnothing$;

2) for each $p \in \mathbf{L}\setminus\{\mathrm{entry},\mathrm{exit}\}$ there are control flows w' and w'' from entry to p and from p to exit respectively.

1.4. Programs Control Flows Graphs

A program control flow graph is used as a program model in the paper.

A program control flow is a transition system $\langle \mathbf{L},\mathbf{T},\mathrm{src},\mathrm{snk}:\mathbf{T}\to\mathbf{L}\rangle$ with maps γ and act. The domain of these maps is the set $\mathbf{T}$ and ranges are the set of assertions and the set of statements respectively. For these maps the following conditions hold:

1) a set of variables for each $\gamma(\tau)$ is a subset of $\mathbf{X}\cup\mathbf{Z}$;

2) if $\mathrm{act}(\tau)=\text{'}v := f(v_1,\ldots,v_s)\text{'}$ and

if $\mathrm{src}(\tau)=\mathrm{entry}$ then $\{v_1,\ldots,v_s\}\subset\mathbf{X}$,

if $\mathrm{src}(\tau)\neq\mathrm{entry}$ then $\{v_1,\ldots,v_s\}\subset\mathbf{X}\cup\mathbf{Z}$,

if $\mathrm{snk}(\tau)\neq\mathrm{exit}$ then $v\in\mathbf{Y}$,

if $\mathrm{snk}(\tau)=\mathrm{exit}$ then $v\in\mathbf{Z}$.

A control flow is a potential protocol of program running. Realizability of such protocol depends on values of input variables, assertions and statements of transitions constituents.

2. Physical Dimensions of Variables

2.1. Dimensions and Dimensional Distributions

Let $\mathbf{U}$ be a finite set. Each element of the set is a name of a physical dimensional unit.

Suggestion 5. Any element from $\mathbf{U}$ corresponds to some basic dimensional unit.

Suggestion 5 provides the unique representation of any physical dimension d in the form $d=\prod_{u\in\mathbf{U}} u^{r_d(u)}$ with independent multipliers [6].

Hence we can consider every dimension as a vector from linear space $\mathbb{Q}^{\mathbf{U}}$ where $\mathbb{Q}$ is the field of rational numbers. In this case each dimension can be represented by vector $\sum_{u\in\mathbf{U}} r_d(u)\mathbf{u}$ where $\mathbf{u}$ is a vector representing the dimensional unit u.

A partial map from $\mathbf{V}$ to $\mathbb{Q}^{\mathbf{U}}$ is called a dimensional distribution on $\mathbf{V}$.

Suppose η is a dimensional distribution. Denote by $[v]^{\eta}$ the value of η on the variable v.

2.2. Dimensional Correctness of Polynomials

Suppose f is a polynomial, η is a dimensional distribution. The aim of the section is to define a set of

η-correctness conditions and η-dimension for the polynomial f.

Denote by $[\![f]\!]$ a set of η-correctness conditions and by $[f]^\eta$ a value of η-dimension for the polynomial f. These concepts are defined by recursion:

1) if $f \equiv v$ then $[\![f]\!] = \varnothing$ and $[f]^\eta = [v]^\eta$;

2) if $f \equiv g \pm h$ then

$$[\![f]\!] = [\![g]\!] \cup [\![h]\!] \cup \left\{ [g]^\eta = [h]^\eta \right\} \text{ and } [f]^\eta = [g]^\eta;$$

3) if $f \equiv g * h$ then

$$[\![f]\!] = [\![g]\!] \cup [\![h]\!] \text{ and } [f]^\eta = [g]^\eta + [h]^\eta;$$

4) if $f \equiv g / h$ then

$$[\![f]\!] = [\![g]\!] \cup [\![h]\!] \text{ and } [f]^\eta = [g]^\eta - [h]^\eta;$$

5) if $f \equiv \mathrm{pow}_n(g)$ then

$$[\![f]\!] = [\![g]\!] \text{ and } [f]^\eta = n \cdot [g]^\eta;$$

6) if $f \equiv \mathrm{root}_n(g)$ then

$$[\![f]\!] = [\![g]\!] \text{ and } [f]^\eta = \frac{1}{n} \cdot [g]^\eta.$$

We shall say that the polynomial f is η-correct if all conditions from the set $[\![f]\!]$ hold.

2.3. Dimensional Correctness of Assertions

Suppose φ is a basic assertion, η is a dimensional distribution.

As above, define the set of η-correctness conditions of the basic assertion φ. It is denoted by $[\![\varphi]\!]$.

If $\varphi \equiv f = g$ where f and g are polynomials then

$$[\![\varphi]\!] = [\![f]\!] \cup [\![g]\!] \cup \left\{ [f]^\eta = [g]^\eta \right\}.$$

Similarly, if $\varphi \equiv f > g$ where f and g are polynomials then

$$[\![\varphi]\!] = [\![f]\!] \cup [\![g]\!] \cup \left\{ [f]^\eta = [g]^\eta \right\}.$$

The set of η-correctness conditions of an arbitrary assertion φ is defined as union of all such sets for its basic constituents.

We shall say that the assertion φ is η-correct if all conditions from the set $[\![\varphi]\!]$ hold.

2.3. Dimensional Correctness of Assigning Statements

Suppose $v := f(v_1, \ldots, v_s)$ is an assignment statement, η is a dimensional distribution.

By definition, put

$$[\![v := f(v_1, \ldots, v_s)]\!] = [\![f(v_1, \ldots, v_s)]\!] \cup \left\{ [v]^\eta = [f(v_1, \ldots, v_s)]^\eta \right\} \quad (10)$$

2.4. Dimensional Correctness of Flows and Programs

Suppose $\langle \mathbf{L}, \mathbf{T}, \mathrm{src}, \mathrm{snk}, \gamma, \mathrm{act} \rangle$ is a model of a program, τ is a transition and η is a dimensional distribution. Denote by $[\![\tau]\!]$ a set of η-correctness conditions for the transition τ. By definition, put

$$[\![\tau]\!] = [\![\gamma(\tau)]\!] \cup [\![\mathrm{act}(\tau)]\!]. \quad (11)$$

Suppose $\tau_1 \ldots \tau_k$ is a control flow for the transition system $\langle \mathbf{L}, \mathbf{T}, \mathrm{src}, \mathrm{snk} \rangle$.

Denote by $[\![\tau_1 \ldots \tau_k]\!]$ a set of η-correctness conditions for the control flow $\tau_1 \ldots \tau_k$.

Evidently,

$$[\![\tau_1 \ldots \tau_k]\!] = \bigcup_{i=1}^{k} [\![\tau_i]\!]. \quad (12)$$

Suppose P is a program and $\langle \mathbf{L}, \mathbf{T}, \mathrm{src}, \mathrm{snk}, \gamma, \mathrm{act} \rangle$ is a model of the program P.

Denote by $[\![P]\!]$ a set of all $[\![w]\!]$ where w is a control flow from entry to exit.

We shall say that program P is η-correct if all sets of conditions from the set $[\![P]\!]$ are satisfied.

A program P is called dimensionally correct iff there exist a dimensional distribution η such that P is η-correct.

3. Dimension Flow Equation

To check dimensional correctness of a program P we build some equation and call it dimension flow equation.

Let $\langle \mathbf{L}, \mathbf{T}, \mathrm{src}, \mathrm{snk}, \gamma, \mathrm{act} \rangle$ be a model of a program.

As usual in static code analysis [7], consider some data lattice. In this case any element of the data lattice is a subset of the set of all subsets of the transitions set $\mathbf{T}$.

For a control flow w denote by $\mathrm{src}(w)$ the element p of $\mathbf{L}$ such that $p = \mathrm{src}(\tau)$, where $w = \tau w'$. Similarly, denote by $\mathrm{snk}(w)$ the element p of $\mathbf{L}$ such that $p = \mathrm{snk}(\tau)$, where $w = w'\tau$.

Let w be a control flow, denote by $\overline{w}$ the set of all transitions generating w.

Let p be an arbitrary element of $\mathbf{L}$, denote by $\mathbf{F}[p]$ the set of all control flows from entry to p. By definition, put

$$W[p] = \{\overline{w} \mid w \in \mathbf{F}[p]\} \tag{13}$$

It is easily shown that for a control flow $w\tau$ the follows equation is satisfied

$$\overline{w\tau} = \overline{w} \cup \{\tau\} \tag{14}$$

Using equations (13) and (14), we get

$$\begin{aligned} W[p] &= \{\overline{w\tau} \mid \tau : snk(\tau) = p, w \in \mathbf{F}[src(\tau)]\} = \\ &\bigcup_{\tau:snk(\tau)=p, w\in\mathbf{F}[src(\tau)]} \{\overline{w} \cup \{\tau\}\} = \\ &\bigcup_{p'\in\mathbf{L}} \bigcup_{w\in\mathbf{F}[p']} \bigcup_{\tau:src(\tau)=p', snk(\tau)=p} \{\overline{w} \cup \{\tau\}\} = \\ &\bigcup_{p'\in\mathbf{L}} \bigcup_{\tau:src(\tau)=p', snk(\tau)=p} \bigcup_{w\in\mathbf{F}[p']} \{\overline{w} \cup \{\tau\}\} = \\ &\bigcup_{p'\in\mathbf{L}} \bigcup_{\tau:src(\tau)=p', snk(\tau)=p} W[p'] * \{\tau\} = \\ &\bigcup_{\tau:snk(\tau)=p} W[src(\tau)] * \{\tau\} \end{aligned} \tag{15}$$

where for $X_i \in \mathbf{2^T}$ $(i = 1,\ldots,m)$, $X \in \mathbf{2^T}$, and denote by $\{X_1,\ldots,X_m\} * X$ the element $\{X_1 \cup X,\ldots,X_m \cup X\}$ of $\mathbf{2^{2^T}}$.

Hence, the map $W : \mathbf{L} \to \mathbf{2^{2^T}}$ satisfies the equation

$$W[p] = \bigcup_{\tau:snk(\tau)=p} W[src(\tau)] * \{\tau\} \tag{16}$$

We say that equation (4) is a dimension flow equation.

For $X : \mathbf{L} \to \mathbf{2^{2^T}}$ by definition put

$$F[X](p) = \bigcup_{\tau:snk(\tau)=p} X[src(\tau)] * \{\tau\} \tag{17}$$

Equation (17) defines the operator F on the lattice of maps from the set $\mathbf{L}$ to the lattice $\mathbf{2^{2^T}}$.

It is easy to prove that the operator F is monotonic.

In these terms equation (17) can be rewritten as

$$X = F[X] \tag{18}$$

and the map W is the least fixed point of the operator F.

Using finiteness of the lattice $\mathbf{2^{2^T}}$, equation (18) and Tarsky's Theorem [8] we can compute the map W.

4. Method for Checking Program Dimensional Correctness

The aim of the section is to describe and ground the method for checking dimensional correctness of a program.

The method has four phases:

1) a program model building;

2) a set of dimensional correctness conditions computing;

3) a set of dimensional correctness conditions simplifying;

4) a set of dimensional correctness conditions checking.

We shall suppose that one can translate program to three-address code [7]. In this case the phase of program model building should provide an evident transformation tree-address code into a program model.

In the second phase the standard least fixed point algorithm [8] is used for computing a set of dimensional correctness conditions. Its correctness is grounded by finiteness of the lattice of maps from the set $\mathbf{L}$ to the lattice $\mathbf{2^{2^T}}$.

The general case of the least fixed point algorithm is shown in Fig. 1.

```
lfp:= proc(
      F: monotonic operator
)
   old:= ∅;
   new:= F[old];
   while new ≠ old do
      old:= new;
      new:= F[new]
   end do;
   return new
end proc
```

Fig. 1. The least fixed point algorithm (general case)

Describe the algorithm for computing the map F defined by formula (17).

Let η be a dimensional distribution defined on $\mathbf{X} \cup \mathbf{Z}$. We can consider the distribution as a dimensional specification of external variables.

After running the least fixed point algorithm we obtain the resulting map W.

The set $W[exit]$ contains some sets of transitions names.

It is easily shown that

1) $W[\text{exit}] = \{\overline{w} \mid w \in \mathbf{F}[\text{exit}]\}$;

2) $W[\text{exit}]$ is a finite set.

```
F:= proc(
     ⟨L,T,src,snk⟩ :
          transition system,
     S : map
)
  for each p∈L do
     U[p]:= S[p];
     for each τ∈T such that
        snk(τ) = p
     do
        for each C∈U[p] do
           delete C from U[p];
           insert C∪{τ}
                      into U[p]
        end do
     end do
  end do;
  return U
end proc
```

Fig. 2. The algorithm to compute the map F

It is evident that $[\![w]\!] = \{[\![\tau]\!] \mid \tau \in \overline{w}\}$, where w is a control flow, and the program is dimensionally correct iff for each $\overline{w} \in W[\text{exit}]$ the conditions set $[\![w]\!]$ is satisfiable.

In the third phase the set $W[\text{exit}]$ is reduced.

To execute this process we shall use the following rule:

if $X \in W[\text{exit}]$, $Y \in W[\text{exit}]$ and $Y \subset X$ then eliminating of the constraints set Y replaces $W[\text{exit}]$ with equivalent set of conditions set.

The proof of correctness the rule is trivial.

In the fourth phase final output of checking process is defined.

Let $\{C_1,\ldots,C_k\}$ be the output of the third phase.

For each $i = 1,\ldots,k$ the set $C_i = \{\tau_{i,1},\ldots,\tau_{i,s_i}\}$ generates the system of conditions

$$\begin{cases} [\![\tau_{i,1}]\!] \\ \cdots\cdots \\ [\![\tau_{i,s_i}]\!] \end{cases} \qquad (19.i)$$

Each subsystem $[\![\tau_{i,j}]\!]$ is a linear system over the field $\mathbb{Q}$. Therefore system (19.i) is a linear system over the field $\mathbb{Q}$ too.

Summing up we can say that to define dimensional correctness of the program P it is necessary and sufficient for all $i = 1,\ldots,k$ to check existence of some solution of system (7.i). There are a lot of methods to do this checking [9].

Conclusion

The method for checking the dimensional correctness of program is built and grounded in the paper.

This method is a method of static source code analysis. Effectiveness of checking the dimensional correctness of a program for its semantic verification was studied in [5]. Experimental results make it possible to evaluate test-sensitivity at the rate of about 65%.

References

1. Littlewood B. Software Reliability and Dependability: A Roadmap / B. Littlewood, L. Strigini // Future of Software Engineering. Limeric, Ireland. – ACM: Proc. FOSE, 2000. – P. 175 – 188.

2. Rushby J.M. Formal Verification of Algorithms for Critical Systems / J.M. Rushby. – IEEE Trans. Soft. Eng. – 1993. – Vol. 19, No 1. – P. 13 – 23.

3. Binkley D. Source Code Analysis: A Road Map / D. Binkley // Future of Software Engineering 2007. – IEEE: Proc. FOSE'07, 2007. – P. 89 – 105.

4. Integrated Instrumental Environment for Critical-Mission Software Systems Assessment, in Russian / B.M. Konorev, V.S. Kharchenko, G.N. Chertkov, Yu.G. Alekseev, Yu.S. Manzhos, V.V. Sergienko. – Kharkiv: I&C System Certification Centre. State Centre for Regulation of Supplies and Service of Quality of State Nuclear Regulatory Committee of Ukraine, 2005. – 258 p.

5. Invariant-Based Assessment of Software for Space Systems, in Russian / B.M. Konorev, V.S. Kharchenko, G.N. Chertkov, Yu.G. Alekseev, Yu.S. Manzhos, V.V. Sergienko. – Kharkiv: I&C System Certification Centre. State Centre for Regulation of Supplies and Service of Quality of State Nuclear Regulatory Committee of Ukraine, 2009. – 221 p.

6. Bridgman P.W. Dimensional Analysis: 2nd ed. / P.W. Bridgman. – New Haven: Yale University Press, 1932. – 116 p.

7. Compilers: principles, techniques, and tools: 2nd ed. / A.V. Aho., M.S. Lam, R. Sethi, J.D. Ullman. – Boston: Pearson Education, Inc., Addison Wesley, 2007. – 1009 p.

8. Ceri S. Logic Programming and Database / S. Ceri, G. Gottlob, L. Tanka. – Berlin: Springer-Verlag, 1990. – 329 p.

9. Poole D. Linear Algebra: A Modern Introduction: 2nd ed. / D. Poole. – Belmont, CA: Thomson Brooks/Cole, 2006. – 559 p.

Поступила в редакцию 20.01.2009

Рецензент: д-р. техн. наук, проф., А.В. Дрозд, Одесский национальный технический университет, Одесса.

ПРО СТАТИЧЕСКИЙ АНАЛИЗ РАЗМЕРНОСТИ ПЕРЕМЕННЫХ ДЛЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КРИТИЧЕСКОГО ПРИМЕНЕНИЯ

С.С. Брюханков, Б.М. Коноров, М.С. Львов, Г.Н. Жолткевич

В работе исследован некоторый сильный метод статического анализа кода, а именно: анализ физической размерности переменных программ. Построена формальная модель для анализа размерности переменных. Размерности и распределения размерностей представлены в статье в целях определения инвариантов для переменных. В статье описан и обоснован метод анализа модели. Показаны экспериментальные результаты, которые позволяют оценить чувствительность теста порядка 65%.

Ключевые слова: надежность программного обеспечения, анализ статического кода, пространственный анализ, поток управления, уравнение потока управления, проблема фиксированной точки.

ПРО СТАТИЧНИЙ АНАЛІЗ РОЗМІРІВ ЗМІННИХ ДЛЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КРИТИЧНОГО ЗАСТОСУВАННЯ

С.С. Брюханков, Б.М. Коноров, М.С. Львів, Г.М. Жолткевич

У роботі досліджено деякий сильний метод статичного аналізу коду, а саме: аналіз фізичної розмірності змінних програм. Побудована формальна модель для аналізу розмірності змінних. Розмірності та розподіл розмірностей розглянуті у статті в цілях визначення інваріантів змінних. У статті описано та обґрунтовано метод аналізу моделі. Показані експериментальні дані що дозволяють оцінити чутливість тесту в 65%.

Ключові слова: надійність програмного забезпечення, аналіз статичного коду, просторовий аналіз, потік керування, рівняння потоку керування, проблема фіксованої точки.

Брюханков Сергей Сергеевич – директор предприятия Mega Business Software Ltd, Украина, e-mail: bss@mbs.kharkov.com.

Коноров Борис Михайлович – д-р техн. наук, проф., проф. каф. компьютерных систем и сетей, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: .admin@scasu.com.

Львов Михаил Сергеевич – канд. физ.-мат. наук, доцент, директор научно-исследовательского института информационных технологий, Херсонский государственный университет, Херсон, Украина, e-mail: lvov@ksu.ks.ua.

Жолткевич Григорий Николаевич – д-р техн. наук, проф., декан механико-математического факультета, Харьковский национальный университет им. В.Н. Каразина, Харьков, Украина, e-mail: zholtkevych@univer.kharkov.ua.