Cybernetics and Systems Analysis, Vol. 42, No. 5, 2006

SYSTEMS ANALYSIS

COMPUTER-BASED TECHNIQUES FOR SOLVING PROBLEMS OF APPLIED AND COMPUTATIONAL MATHEMATICS WITH GIVEN QUALITY CHARACTERISTICS

I. V. Sergienko,[a†] V. K. Zadiraka,[a††] M. D. Babich,[b] A. I. Berezovskii, P. N. Besarab, and V. A. Lyudvichenko[c]

UDC 519.6

The paper is concerned with computer-based techniques for the choice and development of computational resources and their efficient use to find an approximate solution with a given accuracy in a limited processor time.

Keywords: *computer-based techniques, mathematical model of applied problem, computational model, computational process, computational algorithm, estimates of quality characteristics of computational algorithm.*

INTRODUCTION

The present paper continues [1–5] and is concerned with development of computer-based techniques (CT) for solving problems of applied and computational mathematics with preset quality characteristics (i.e., choice and construction of computational resources) and their efficient use for finding an approximate solution with a prescribed accuracy in a limited CPU time.

A general framework to solve problems of applied and computational mathematics using computers includes the following steps.

1. Formulation of an applied problem in terms of an application domain.
2. Choice of a mathematical model (MM) for the applied problem.
3. Choice of a computer-based computational model (CM) that would include the following components:
   - initial data on the problem;
   - class of problems of computational mathematics based on the initial data;
   - class of computational algorithms (c.a.) to calculate the solution, estimate the quality characteristics and the parameters of the computational process (c.p.).[1];
   - computer architecture;
   - software for the c.p.;
   - constraints for quality characteristics.
4. Possible adjustments of the MM and components of the computational model and reexamination of steps of the solution framework.
5. Designing the c.p. and conducting computations.
6. Interpretation of the computed results.

[1]By a computational process, we mean problem-solving process that uses software and hardware to transform initial data into solution results.

[a]V. M. Glushkov Institute of Cybernetics, National Academy of Sciences of Ukraine, Kyiv, Ukraine, [†]*aik@public.icyb.ua*, [††]*zvk140@ukr.net*. [b]2Ukrainian Academy of Foreign Trade, Kyiv, Ukraine. [c]International Academy of Human Resource Management (MAUP), Kyiv, Ukraine. Translated from Kibernetika i Sistemnyi Analiz, No. 5, pp. 33-41, September-October 2006. Original article submitted June 8, 2006.

1060-0396/06/4205-0641

The general solution framework generates a set of CTs, depending on the degree of development and specific use of the above steps. The factors generating the set of CTs include the type of the problem and the MM, initial data available (form, volume, and accuracy), requirements to the approximate (numerical) solution and constraints for computer resources (CPU time and memory), computer capabilities, algorithms and software available, and qualification of programmers and users.

Since we consider CTs, the emphasis is on steps 3–5.

Step 2 is also associated (to some extent) with CTs.

Consider a formulation of an applied problem in terms of the corresponding application domain (i.e., target, information needed, accuracy). Assume that quantitative information is ordered (or given) in a certain system of units (for example, SI). This applied problem is associated with some MM. Since MM is used in CT to solve the applied problem, certain requirements are imposed on the MM. The MM will be used in the CT in the form of a sequence of problems of computational mathematics, which are to be solved (as a rule) approximately by numerical algorithms under certain requirements for quality characteristics of the approximate solution and c.p. under restricted computational resources. Numerical stability and performance of computational algorithms are of importance. It is also important to know whether efficient software for numerical implementation of the MM is available and there is experience of its use. If one or both conditions fail, it may be expedient to refer to another MM, which will make it possible to derive an approximate solution with better quality characteristics. This situation can be illustrated in the following way. The Cauchy problem for ordinary differential equations pertains to problems for which a wide range of software is developed (because of specificity of different classes of equations and features of numerical implementation of corresponding programs on computers of different architectures). There is a wide experience of using software, i.e., possibilities to solve efficiently Cauchy problems of various classes. In this connection, situations are known where other problems are reduced to Cauchy problems [6]. Generally speaking, it is not always possible to select a priori the best (in a sense) MM; therefore, the problem of choosing MM can be considered at some stage of designing a CT. The expediency of such an approach depends on the applied problem and its solution with a given criterion. For example, the mathematical model of some phenomenon will be used repeatedly for values of model parameters within a rather narrow range. Thus, in the case of a complete simulation of the phenomenon, even considerable efforts to optimize the computational complexity of the model can be justified. The error of the MM is not considered; however, constraints for the measure of the total error of approximate solution is assumed to include the error of the MM.

COMPUTATIONAL MODEL. FORMULATION OF THE PROBLEM

Let F be a class of problems of computational mathematics, $A(X)$ be a class of computational algorithms (c.a.) to solve problems of the class F using initial information $I = I(I_0, I_n(f))$, where I_0 is information on the properties of problems from the class F, $I_n = I_n(f) = (i_1(f), \ldots, i_n(f))^T$ is information on the problem f in the form of n functionals calculated on elements of the problem f.

Let $c(Y)$ be a computer model that includes certain architectural properties of the computer and belongs to a class of models $C(Y)$.

The following problem is set: compute a solution (approximate in the general case) to a problem $f \in F$ under the conditions

$$\rho(E(I, X, Y)) \le \varepsilon, \tag{1}$$

$$T(\varepsilon, I, X, Y) \le T_0(\varepsilon), \tag{2}$$

$$M(\varepsilon, I, X, Y) \le M_0, \tag{3}$$

where $\rho(\cdot)$ is a measure of the error of the approximate solution to the problem $f \in F$; $E(I, X, Y)$ is, as a rule, the total error of the approximate solution, which is the sum of three components: the irreducible (inherent) error $E_H(\cdot)$ due to the inaccuracy of the initial data, the method error $E_\mu(\cdot)$, and the round-off error $E_\tau(\cdot)$ [1]; X and Y are parameter vectors that characterize algorithms and computers from the classes A and C, respectively; $T(I, X, Y)$ and $M(I, X, Y)$ are the CPU time and computer memory necessary to compute the approximate solution; ε, $T_0(\varepsilon)$, and $M_0(\varepsilon)$ are the constraints based on requirements to mathematical simulation and properties of the initial information (volume, accuracy, structure, and the acquisition method).

The approximate solution for which condition (1) holds is called an ε-solution, $A(\varepsilon, X, Y)$ is the set of c.a. of obtaining the ε-solution in the given CM.

A computational algorithm satisfying conditions (1) and (2) is called T-efficient, $A(\varepsilon, T_0, X, Y)$ is the set of T-efficient c.a. in the given CM [4].

Hereafter, we assume (if there are no other assumptions) that the memory M can be expanded to a desired size (i.e., constraint (3) can be omitted) but, probably, at the expense of increasing CPU time, for example, by expanding a certain type of memory in the general structure of the computer memory. Since $\varepsilon \to 0$ yields $M_0(\varepsilon) \to \infty$ (for example, when a round-off error or a method error in recurrent algorithms is dealt with), it is agreed that $\varepsilon \geq \varepsilon_0$, where ε_0 is a given number.

SOME GENERAL PROBLEMS OF DESIGNING CT

Let ε^0 be the lower bound of the error of an approximate solution and $T^0(\varepsilon)$ be the lower bound of the CPU run time for the ε-solution in the given CM [6, 7]. Depending on computer resources and constraints (1) and (2), we may distinguish the following situations with respect to the sets $A(\varepsilon, X, Y)$ and $A(\varepsilon, T_0, X, Y)$:

$$A(\varepsilon, X, Y) \neq \varnothing,\ (\varepsilon \geq \varepsilon^0),\ A(\varepsilon, T_0, X, Y) \neq \varnothing,\ (T_0(\varepsilon) > T^0(\varepsilon)), \tag{4}$$

$$A(\varepsilon, X, Y) = \varnothing,\quad (\varepsilon < \varepsilon^0), \tag{5}$$

$$A(\varepsilon, X, Y) \neq \varnothing,\ (\varepsilon \geq \varepsilon^0),\ A(\varepsilon, T_0, X, Y) = \varnothing,\ (T_0(\varepsilon) < T^0(\varepsilon)). \tag{6}$$

Conditions (4)–(6) characterize designing a c.p. under given computational conditions and involve the following problems [4]:

— existence of the ε-solution (possibility of deriving an ε-solution in the given CT);

— existence of T-efficient c.a.;

— adjusting CT or passing to a new one in cases (5) and (6) in order to reduce these situations to situation (4);

— designing a real c.p.

These problems can be solved by choosing components of the CM and the way of their efficient use. Estimates for the measure of the error of the approximate solution and CPU time play an important role. Below we will consider the specific features of applying these quality characteristics.

STRUCTURE OF THE TOTAL ERROR. ACCURACY AND COMPUTATIONAL COMPLEXITY

The computational complexity (T) is frequently defined by requirements to the accuracy of an approximate solution, relationships among components of the total error, dependence of the error on type, structure, and amount of initial data and their accuracy, computer word length and round-off rule, and on the type of error estimates. These circumstances are accounted for in a definite way in forming a computational model and its use to design a c.p. Let us consider how the contribution of each component in the total error influences conditions (1) and (2). The information I_0 (that determines a class of problems) and $I_n(f)$ (that pertains to a specific problem), as a rule, is given approximately. Let I_0^T and $I_n^T(f)$ be exact values of this information, and the information $I_0, I_n(f)$ be exact for some problem φ; R_f and R_φ be exact solutions for the information $I_0^T, I_n^T(f)$ and $I_0, I_n(f)$, respectively; R_f^α and R_φ^α are approximations to R_f and R_φ, respectively, calculated by some algorithm on the assumption that the calculations are exact (without round-off); and R_f^τ and R_φ^τ are round-off solutions by the same algorithm. Then

$$E(I, X, Y) = R_f - R_\varphi^\tau = (R_f - R_\varphi) + (R_\varphi - R_\varphi^\alpha) + (R_\varphi^\alpha - R_\varphi^\tau)$$

$$= E_H(\cdot) + E_\mu(\cdot) + E_\tau(\cdot),$$

$$\rho(E) \leq \rho(E_H) + \rho(E_\mu) + \rho(E_\tau).$$

Let us consider possible distributions of terms in the condition $\rho(E_H)+\rho(E_\mu)+\rho(E_\tau)\le\varepsilon$ with condition (2) taken into account. Let us replace this condition with the following ones:

$$\rho(E_H)\le\delta_H,\ \rho(E_\mu)\le\delta_\mu,\ \rho(E_\tau)\le\delta_\tau,$$

where $\delta_i\le\alpha_i\varepsilon$, $\sum\alpha_i=1$, $\alpha_i\ge0$, and $i=H,\ \mu,\ \tau$.

The above-mentioned distribution is determined by the choice of numbers $\{\alpha_i\}$. Unsuccessful distribution may substantially complicate conditions (1) and (2). For example, if $0\le\varepsilon-\rho(E_H)<<\varepsilon$, it is necessary to impose stringent constraints on two other components of the total error:

$$\rho(E_\mu)+\rho(E_\tau)\le\varepsilon-\rho(E_H)<<\varepsilon. \tag{7}$$

Let, for example, c.a. implement numerical integration algorithm (integrals and differential equations), where $I_n(f)$ is a set of values of the integrand or right-hand sides of equations. In this case, for a definite round-off rule,

$$E_\mu(I_n(f))=O(n^{-p}),\ E_\tau(I_n(f))=O(n2^{-\tau}),$$

where p is the order of accuracy of the numerical method and τ is the mantissa length for base 2 numbers. The following relation holds for the measure of computational error $\rho^0_{\mu\tau}$:

$$\rho^0_{\mu\tau}(\tau)=\min_n(E_\mu+E_\tau)=O(n_0^{-p}(\tau)), \tag{8}$$

where $n_0=O(2^{\tau/(p+1)})$, $\rho(E_\mu),\rho(E_\tau)=O(2^{-\tau p/(p+1)})$, and $\tau\to\infty$.

As is seen from (8), the minimum of the measure of the computational error $\rho^0_{\mu\tau}(\tau)$ is achieved for each τ for some $n_0(\tau)$. Necessary existence condition for the ε-solution $\rho^0_{\mu\tau}(\tau)\le\delta=\varepsilon-\rho(E_H)$ is related to

$$n=O(\delta^{-1/p}),\quad \tau=O(\log\delta^{-1}),\ \delta\to0. \tag{9}$$

To calculate ε-solution using a linear computational scheme, the CPU time

$$T(\varepsilon)=n(\delta)\beta(\delta)\alpha(\tau) \tag{10}$$

is required, where $\beta(\delta)$ is the average number of operations to calculate one functional from the set $I_n(f)$ and its use for the c.a., and $\alpha(\tau)$ is the average cost of one operation in calculating ε-solution.

If we put $\alpha(\tau)=O(\tau)$ (in parallelizing multiplication of two numbers) [8], then we get from (9) and (10) $T(\varepsilon)=O(\delta^{-1/p})\log\delta^{-1}$, $\delta\to0$. Thus, situation (6) can follow from condition (7).

TYPES OF PROBLEMS IN VIEW OF COMPUTATIONAL COMPLEXITY AND CHARACTERISTIC ESTIMATES

Possibilities of solving the problem under conditions (1)–(3) may significantly depend on the used type of error estimates.

As is generally known [1], estimates can be a priori and a posteriori, majorizing and asymptotic, deterministic and stochastic. Each of the types of estimates has its benefits and disadvantages. For example, a priori majorizing deterministic estimates, on the one hand, are guaranteed, easily calculated, require no problem to be solved, and on the other hand, an estimate may be highly overstated and of little use for quantitative analysis. The situation remains the same in the class of problems, even if the estimate is optimal. The problem in which this estimate is achieved may be not typical of the class; therefore, capabilities of the c.a. remain untapped for other problems of the class.

Asymptotic a posteriori estimates can be sufficiently close to the quantity under study; however, such a proximity is achieved for values of the parameter (being varied) from a certain domain of asymptotics, which is not always convenient for practical computations. Moreover, to compute such estimates, the problem solution is used, which needs substantial amount of calculations.

A computational situation determines what type of estimates is expedient to use. Let, for example, the c.p. parameters X and Y for which condition (1) holds be selected for numerical integration based on (1). If the majorizing estimates are

used, then n (the number of the integrand values) can be much greater than that for which the ε-solution can be calculated, which increases the CPU time (T). Such a situation may be absolutely acceptable for problems of small computational complexity.

Let us distinguish groups of problems whose solution, under conditions (1)–(3), may reasonably employ asymptotic a posteriori estimates as exactest ones, despite their significant computational complexity. Such problems include the following:

— problems (or series of problems) to be solved in real time;

— large series of equitype problems (parameters vary in a rather narrow range; for example, a functional minimization problem, where points of the phase space are solutions to systems of differential or integral equations);

— problems solved by c.a.-programs [3] from problem-oriented software libraries (for programs to be efficiently used, it is desirable to know exact characteristics of typical representatives of the class of problems);

— problems with much calculation to be solved in a practically reasonable time;

— problems with highly accurate calculations.

RESERVES FOR IMPROVING THE QUALITY CHARACTERISTICS OF THE SOLUTION AND C. P.

Reserves for decreasing the measure of errors:

a) initial data ($\rho(E_H)$):

- improvement of the class of problems;
- adjustment of the initial information;
- increasing accuracy of the initial information;

b) method ($\rho(E_\mu)$) :

- use of optimal c.a. and c.a. optimal in the order of accuracy;
- optimization of the information set of functionals (for example, an optimal mesh for numerical integration);
- increase in the number of functionals in the information set;
- passage to another class of information operators, which provides a greater optimal order of c.a. than the given class of information operators does;
- full use of the initial information to narrow the class of problems;

c) round-off ($\rho(E_\tau)$):

- using methods of optimal order of accuracy for exact calculations;
- narrowing the class of problems (to decrease the number of functionals in the information set);
- using computational schemes that minimize the velocity of round-off error accumulation;
- increasing the optimal order of accuracy by selecting the class of information operators;
- increasing the word length;
- choice and simulation of the round-off rule;
- using optimized information set of functionals.

Reserves for decreasing the CPU time:

- improving the class of problems;
- using methods of optimal order of accuracy;
- using specified initial information (which results in less stringent constraints for the method and round-off errors);
- improving quality of estimates of the method and round-off errors;
- increasing calculation accuracy for the c.p. parameters;
- coordinating the c.a. with the computer architecture;
- “fast” arithmetics [8, 9];
- parallelizing calculations;
- developing specialized calculators (choice of the computer architecture that would agree best with the c.a. to solve problem of the given class).

ELEMENTS OF THE TECHNIQUE OF SOLVING PROBLEMS WITH GIVEN VALUES OF QUALITY CHARACTERISTICS

To solve a problem with preset quality characteristics (1)–(3), one should know what to do, how, and in what sequence.

Let us briefly describe the sequence of steps of a technique implementing the steps of the above general framework of problem solution.

Step 1. Based on the initial data, assign the MM of the applied problem f to a certain class of problems F of computational mathematics. Analyze the information $I = I(I_0, I_n(f))$ on elements of problem formulation and its desired solution (their quantitative values and properties: initial data of the problem, estimates of the accuracy of their representation (or calculation), etc.).

Step 2. Based on practical needs (technological requirements of the modeled system, process, or phenomenon) and the accepted system of units (SI as a rule) in the mathematical model of the applied problem, after a corresponding scaling (if necessary), impose requirements (1)–(3) on the quality characteristics of the desired solution: the accuracy $(\varepsilon > 0)$; computer time $T_0(\varepsilon)$ of deriving an ε-solution; and the computer memory M_0 to be satisfied by the corresponding c.a.-program.

Step 3. Using initial data $I = (I(I_0, I_n(f))$ on the problem $f \in F$ and estimates of the accuracy of their representation (calculation), find the estimate E_H of the irreducible (inherent) error of the desired solution to the problem f.

Step 4. For the found estimate E_H of the irreducible error of the desired solution, check fulfillment of the condition

$$\rho(E_H) < \alpha_1 \cdot \varepsilon, \quad 0 < \alpha_1 < 1, \quad \text{for example,} \quad \alpha_1 = 1/3. \tag{11}$$

If (11) holds according to the prescribed accuracy (1) for the desired approximate solution of the problem f, then go to Step 6, otherwise go to Step 5.

Step 5. To solve the problem $f \in F$ with accuracy (1), change (if possible) the requirement to the accuracy of the desired solution (increase ε) or reduce (improve) the estimate of the irreducible error E_H by using reserves for its improving. Go to Step 2 or 3, respectively. If $\rho(E_H)$ considerably exceeds the given ε and cannot be reduced (if the problem being solved is incorrect [10]), change formulation of the problem $f \in F$ and go to Step 1.

Step 6. Based on the a priori information $I = I(I_0, I_n(f))$ on the problem $f \in F$ being solved and requirements to values of the quality characteristics of the desired solution (1)–(3), separate out or introduce a new subclass $\widetilde{F}$ of problems in the class F. The narrower the subclass $\widetilde{F} \subset F$ into which the problem is immersed (i.e., the more the a priori information used to solve the problem), the greater accuracy and performance of solving the problem are possible.

Step 7. Step 6 leads to the following situations:

1) The method to solve the subclass of problems $\widetilde{F}$ is not known; therefore, the set of c.a. has not been developed and estimates of their characteristics have not been analyzed theoretically; go to Step 8;

2) Methods to solve the subclass of problems $\widetilde{F}$ are known, but c.a. and software for a corresponding computer $c(Y)$ have not been developed and estimates of their characteristics (E, T, M) have not been analyzed theoretically; go to Step 9;

3) A software system (for example, Mathematica, Matlab or Mathcad) or an application package are available to solve the class of problems F that includes a c.a.-program for the solution of a subclass of problems $\widetilde{F} \subset F$; but it has no estimates of characteristics (E, T, M), which may be used to judge whether it is possible to solve the problem $f \in \widetilde{F}$ with required values of the quality characteristics (1)–(3); go to Step 15;

4) For the class of problems F, an application package is available, which contains a c.a.-program to solve a subclass of problems $\widetilde{F} \subset F$; a feature of this package is that a c.a.-program for solving the problem $f \in \widetilde{F}$ has estimates of characteristics E, T, M and finds ε-solution of the problem $f \in \widetilde{F}$ with a prescribed (within some range $D(\varepsilon)$) accuracy $\varepsilon \in D(\varepsilon)$; go to Step 16.

Step 8. No method is available (or has been developed) for the subclass of problems $\widetilde{F}$ to which the applied problem f being solved belongs; therefore, develop one.

To develop a method, substantiate it, and analyze its characteristics (design a theory and prove associated theorems), one can either use well-known approaches or develop new ones to solve the subclass of problems $\widetilde{F}$.

Step 9. From the set of well-known (or developed according to Step 8) methods of solving the class of problems F or $\widetilde{F}$ using the known a priori estimates of characteristics (for example, estimates of the method error, convergence rate,

information and combinatory complexity), select the "best" one in terms of accuracy and performance, which can be used to solve the class F or the subclass $\widetilde{F} \subset F$.

Step 10. Choose parameters of the computer to solve the problem $f \in \widetilde{F}$ (based on the initial analysis of its complexity and requirements to the quality characteristics of the solution), i.e., define a computer model $c(Y) \subset C(Y)$ by choosing parameters Y: the number of processors ($k = 1$ or $k > 1$) and their type, word length, the number of machine words to be used for arithmetic operations, estimates of the time to perform operations, and constraints M_0 for the RAM used, etc.

Step 11. Use the developed or selected method to develop a T-efficient c.a. for ε-solution of the problem $f \in \widetilde{F}$ and estimate characteristics of the c.a. $E(I, X, Y)$, $T(I, X, Y)$, and $M(I, X, Y)$ according to the technique described in [1, 5].

Step 12. Implement the developed T-efficient c.a., for ε-solution of the problem $f \in \widetilde{F}$, in the c.a.-program in certain programming language (for example, Pascal or C) for the selected computer $c(Y) \subset C(Y)$.

The c.a.-program being developed, which solves the problem $f \in \widetilde{F}$ with required values of the quality characteristics (1)–(3), should belong to one of the following types of c.a.-programs:

• c.a.-program for computing ε-solution with required accuracy: the values of its control parameters X, to provide required accuracy (1) from its accuracy range $D(\varepsilon)$, $\varepsilon \in D(\varepsilon)$, are calculated in the program, using a priori estimates of accuracy; components of the vector X of control parameters of the c.a.-program may be, for example, the number of iterations, word length, etc.

• c.a.-program for computing ε-solution to the problem and a posteriori estimate of the error of the obtained solution: the c.a.-program finds ε-solution of the problem with required admissible accuracy $\varepsilon \in D(\varepsilon)$ by proper selection of its control parameters X, with the use of a posteriori estimate of the accuracy of the approximate solution being computed.

• c.a.-program for computing approximate solution of the problem $f \in \widetilde{F}$ (without computing accuracy estimate): the user can provide the required accuracy $\varepsilon \in D(\varepsilon)$ by appropriately choosing control parameters X of the c.a.-programs, using the accuracy estimates obtained during its testing.

Step 13. Use appropriate set of test problems to test the c.a.-program developed for the solution of the considered class of problems and estimates of the obtained characteristics E according to the technique of testing quality characteristics of c.a.-programs [3].

If testing results confirm that the c.a.-program developed can find solutions of the problem $f \in \widetilde{F}$ with required values of the quality characteristics (1)–(3), then go to Step 17, otherwise go to Step 14.

Step 14. Analyze the c.a.-program to reveal reserves for its improvement (optimization) from the characteristics that do not meet the prescribed requirements; optimize the c.a.-program, operate with the obtained modification of the c.a.-program according to the technique [4]; if all optimization reserves for the selected c.a. and the program implementing it are exhausted but its characteristics still do not satisfy the set of constraints (1)–(3), continue searching for the corresponding c.a. according to the previous steps (i.e., go to Steps 10 and 11).

Step 15. Test the c.a.-program selected from a software system to solve the class of problems $\widetilde{F}$ and estimate its characteristics: accuracy of the problem solution E, run time T, required computer memory M according to the testing technique [3].

If the testing results confirm that the selected c.a.-program can solve the problem $f \in \widetilde{F}$ with required values of the quality characteristics in accuracy and performance, go to Step 17, otherwise go to Step 7.

Step 16. Using the estimates (available in the application package) of E, T, and M, check whether it can find ε-solution of a problem $f \in \widetilde{F}$ with accuracy $\varepsilon \in D(\varepsilon)$ prescribed by (1) and for a given constraint (2) for machine time. If this c.a.-program finds ε-solution of the problem $f \in \widetilde{F}$ with prescribed quality characteristics (1)–(3), go to Step 17, otherwise to Step 7.

Step 17. Design (using a c.a.-program developed or selected for a subclass of problems $\widetilde{F}$) the solution of a problem $f \in \widetilde{F}$ with prescribed values of quality characteristics (1)–(3).

REFERENCES

1. V. V. Ivanov, M. D. Babich, A. I. Berezovskii, P. N. Besarab, V. K. Zadiraka, and V. A. Lyudvichenko, Characteristics of Problems, Algorithms, and Computers in Software Systems of Computational Mathematics [in Russian], V. M. Glushkov Inst. Cybern., NAS USSR, Kyiv (1984).
2. V. S. Mikhalevich, I. V. Sergienko, V. K. Zadiraka, and M. D. Babich, "Optimization of computations," Cybern. Syst. Analysis, Vol. 30, No. 2, 213–235 (1994).
3. M. D. Babich, V. K. Zadiraka, and I. V. Sergienko, "A computing experiment in the problem of optimization of computations, Pt. I, Pt. II," Cybern. Syst. Analysis, Vol. 35, No. 1, 47–55; No. 2, 221–239 (1999).
4. M. D. Babich, A. I. Berezovskii, P. N. Besarab, V. K. Zadiraka, V. A. Lyudvichenko, and I. V. Sergienko, "*T*-efficient calculation of the ε-solutions to problems of calculus and applied mathematics, Pt. I, Pt. II," Cybern. Syst. Analysis, Vol. 37, No. 2, 187–202; No. 3, 381–397 (2001).
5. V. K. Zadiraka, M. D. Babich, A. I. Berezovskii, P. M. Besarab, L. O. Gnativ, and V. O. Lyudvichenko, *T*-Efficient Algorithms for Approximate Solution of Problems of Calculus and Applied Mathematics. Scientific Edition [in Ukrainian], V. M. Glushkov Inst. Cybern. NASU (2003).
6. V. V. Ivanov, Methods of Computations: A Reference Book [in Russian], Naukova Dumka, Kyiv (1986).
7. J. F. Traub and H. Wozniakovski, A General Theory of Optimal Algorithms, Academic Press, New York (1980).
8. M. A. Kartsev, Arithmetics of Digital Computers [in Russian], Nauka, Moscow (1969).
9. V. K. Zadiraka and O. S. Oleksyuk, Computer Arithmetics of Multidigited Numbers [in Ukrainian], V. M. Glushkov Inst. Cybern., NAS USSR, Kyiv (2003).
10. A. N. Tikhonov and V. Ya. Arsenin, Methods to Solve Ill-Posed Problems [in Russian], Nauka, Moskow (1986).