ISSN 0965-5425, Computational Mathematics and Mathematical Physics, 2010, Vol. 50, No. 12, pp. 2166–2174. © Pleiades Publishing, Ltd., 2010.
Original Russian Text © M.D. Babich, V.K. Zadiraka, V.A. Lyudvichenko, I.V. Sergienko, 2010, published in Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 2010, Vol. 50, No. 12, pp. 2285–2295.

Dedicated to Academician A.A. Dorodnicyn on the Occasion of the Centenary of His Birth

On the Use of Computation Optimization Opportunities in Computer Technologies for Applied and Computational Mathematics Problems with Prescribed Quality Characteristics

M. D. Babich, V. K. Zadiraka, V. A. Lyudvichenko, and I. V. Sergienko

Glushkov Institute of Cybernetics, National Academy of Sciences of Ukraine, pr. Glushkova 40, Kiev, 03187 Ukraine

e-mail: myhailo.babich@gmail.com, zvk140@ukr.net, ljudvo@voliacable.com, aik@public.icyb.kiev.ua

Received May 20, 2010

Abstract—The use of various opportunities for computation optimization in computer technologies for applied and computational mathematics problems with prescribed quality characteristics is investigated. More precisely, the choice and determination of computational resources and methods of their efficient use for finding an approximate solution of problems up to prescribed accuracy in a limited amount of processor time are investigated.

DOI: 10.1134/S0965542510120171

Keywords: computer technology, mathematical model of applied problem, model of computations, computation procedure, computational algorithm, estimates of computational algorithm quality.

1. INTRODUCTION

At every phase of the development of computer capacities and methods for solving applied problems, superlarge-scale problem arise (like controlling fast processes, processing of radar data, diagnostics of reactors, determining the maximum acceptable level of methane in mines, etc.). To overcome the superexponential computational complexity barrier, it is important to use various opportunities to optimize computations both by developing solution methods that are optimal in terms of accuracy and computation time and by creating more powerful computers. It is known that, for certain classes of problems (digital signal processing, pattern recognition, and the like), the effect of using optimal methods and parallel computing is comparable with the effect of using more powerful hardware.

In this paper, we describe the results of studies in the field of optimization of computations as applied to the approximate solution of computational and applied mathematics problems carried out in the Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine.

The generic scheme of the solution of various computational and applied mathematics problems using computer technologies (see 1–3]) involves the following stages.

1. Statement of an applied problem in terms of the application field.

2. Selection of a mathematical model (MM) of the applied problem.

3. Selection of a computation model (CM), which includes the following components:

—initial data of the problem;

—a class of computational mathematics problems based on the initial data;

—a class of computational algorithms (CAs) for finding a solution, constructing estimates of quality characteristics and parameters of the computation procedure (CP)[1];

—computer architecture;

—restrictions on the values of the quality characteristics.

4. Modification of the MM, components of the CM, and the repeated application of the stages of the scheme.

5. Construction of CP and actual computations.

6. Interpretation of the results.

[1] By a CP, we mean a procedure of solving a problem based on processing the initial data so as to produce a solution using computer software and hardware.

This generic scheme can produce many computer technologies depending on the thoroughness of using the stages listed above. The factors that affect the set of computer technologies produced by this scheme are as follows: the problem type and the MM, the available initial data (their form, amount, and accuracy), requirements for the approximate (numerical) solution of the problem and restrictions on the computation resources (CPU time, computer memory), capabilities of computers, the available algorithms and software, and the skills of the developers and software users.

Since we consider computer technologies in this paper, we focus on stages 3–5 of the above scheme. Stage 2 is also related to computer technologies to a certain degree.

Suppose that we have a statement of an applied problem in terms of the corresponding application field (that is, we assume that there is a description of what should be found and which information (and its accuracy) can be used for that purpose). We assume that the quantitative information is given in a certain system of units (e.g., in SI). To such an applied problem, we assign a MM. In order to use this MM for the construction of a computer technology, we impose certain requirements upon the MMs. Based on the MM, a sequence of computational mathematics problems will be obtained in the computer technology that must be solved using numerical algorithms under certain requirements for the quality characteristics of the approximate solution and for the CP under limited computation resources.

The issues of numerical stability and performance of computational algorithms are also important. It is essential to know which software is available for the numerical implementation of the MM and whether there are positive references of its practical usage. If one or both of these conditions are not fulfilled, it may be reasonable to use another MM that enables one to obtain an approximate solution with better quality characteristics. Generally speaking, it is not always possible to select the best (in a certain sense) MM a priori; therefore, the problem of selecting an MM can be considered as the most important one at a certain stage of constructing the computer technology. The efficiency of using such an approach depends on the applied problem and on its solution under the prescribed quality criterion for the approximate solution. For example, if the MM of a certain phenomenon is supposed to be used repeatedly with the parameter values that vary in a certain range, even significant efforts for the optimization of the computational complexity of the model can be warranted. We do not consider the possible errors in the MM; however, we assume that the restrictions on the acceptable total error in the approximate solution are chosen with regard to the error of the MM.

2. COMPUTER MODEL OF COMPUTATIONS AND GENERAL ISSUES OF COMPUTER TECHNOLOGIES CONSTRUCTION

Let K be a class of computational mathematics problems and $A(X)$ be a class of computational algorithms designed for solving problems in the class K; these algorithms use the initial data $I = I(I_0, I_n(f))$, where I_0 is some information about the properties of the problems in K and $I_n = I_n(f) = (i_1(f), ..., i_n(f))^{\mathrm{T}}$ is some information about the problem f in the form of n functionals calculated on the elements of f.

Let $c(Y)$ be a model of a computer that takes into account the architectural features of the computer and belongs to a class of models $C(Y)$.

The problem is to find a solution (generally, an approximate one) of the problem $f \in K$ subject to 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 error in the approximate solution of the problem $f \in K$; I is the initial data of the problem; $E(I, X, Y)$ is typically the total error of the approximate solution, which is the sum of three components—the irremovable error due to the errors in the initial data $E_H(\cdot)$, the error due to the method (algorithm) $E_\mu(\cdot)$, and the rounding error $E_\tau(\cdot)$ (see [4–6]); X and Y are the vectors of parameters that characterize the algorithms and computers in the classes A and C, respectively; $T(I, X, Y)$, $M(I, X, Y)$ are the CPU time and the computer memory needed to calculate the approximate solution; ε, $T_0(\varepsilon)$, and $M_0(\varepsilon)$ are the restrictions based on the requirements for the quality of the mathematical modeling and the properties of the initial data (amount, accuracy, structure, and the acquisition method).

An approximate solution satisfying condition (1) is called an ε-solution, and $A(\varepsilon, X, Y)$ is the set of computational algorithms designed for finding ε-solutions in the CM under examination.

A computational algorithm satisfying conditions (1) and (2) is said to be T-efficient; we denote by $A(\varepsilon, T_0, X, Y)$ the set of T-efficient algorithms in the CM under examination (see [7]).

Below, we assume that the memory M can be extended up to the desired size; that is, we assume that condition (3) can be removed (possibly, at the cost of increasing the CPU time), for example, by extending the memory of a certain type in the structure of computer memory. Since $\varepsilon \longrightarrow 0$ as $M_0(\varepsilon) \longrightarrow \infty$ (e.g., when the rounding errors or the errors inherent in the method in recurrent algorithms are discussed), we assume that $\varepsilon \geq \varepsilon_0$, where ε_0 is a given number.

Let ε^0 be a lower bound of the error of the approximate solution and $T^0(\varepsilon)$ be a lower bound of the CPU time of calculating an ε-solution in the given CM (see [5, 8]). Depending on the computer resources and conditions (1), (2), one can distinguish the following situations concerning the sets $A(\varepsilon, X, Y)$ and $A(\varepsilon, T_0, X, Y)$:

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

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

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

Conditions (4)–(6) characterize the construction of a CP under the given computation conditions. They are connected with the following issues (see [7]):

—existence of an ε-solution (the possibility to construct an ε-solution using the given computer technology);

—existence of T-efficient computational algorithms;

—modification of the computer technology or its replacement with another one in cases (5) and (6) in order to reduce these situations to situation (4);

—construction of an actual CP.

These issues are resolved by selecting the components of the CM and a way to use them efficiently. Here, an important role belongs to estimates of the error of the approximate solution and of the CPU time. Below, we consider some particulars of the application of these quality characteristics.

2.1. Structure of the Total Error, Accuracy, and Computational Complexity

The computational complexity (T) is often determined by the requirements for the accuracy of the approximate solution, relationships between the components of the total error, the dependence of the error on the type, structure, and size of the initial data and on their accuracy, the computer word size and rounding rules, and on the type of error estimates. Therefore, it is reasonable to consider the two characteristics (the error of the approximate solution and the CPU time) together.

Consider the influence of the components of the total error on the fulfillment of conditions (1), (2). The information I_0 (which determines the class of problems) and $I_n(f)$ (which depends on a particular problem) is typically given only approximately. Let I_0 and $I_n(f)$ be the exact information and I_0^T, $I_n^T(f)$ be exact for a certain problem φ; let R_f and R_φ be exact solutions of the problems corresponding to the information I_0^T, $I_n^T(f)$ and I_0, $I_n(f)$, respectively. Furthermore, assume that R_f^α, R_φ^α are the approximations of R_f and R_φ, respectively, found by a certain algorithm under assumption that all the computations are performed exactly (without roundoffs), and let R_f^τ, R_φ^τ be the solutions found by the same algorithm with regard to the rounding. Then, we have

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

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

Consider possible distributions of the terms in the expression (bound)

$$\rho(E_H) + \rho(E_\mu) + \rho(E_\tau) \leq \varepsilon$$

with account of condition (2). We replace this bound with the bounds

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

where

$$\delta_i \le \alpha_i \varepsilon, \quad \sum \alpha_i = 1, \quad \alpha_i \ge 0, \quad i = H, \mu, \tau.$$

The distribution of the terms is determined by the choice of the numbers $\{\alpha_i\}$. A poor distribution can considerably complicate the fulfillment of conditions (1), (2). For example, when $0 \le \varepsilon - \rho(E_H) \ll \varepsilon$, stringent restrictions upon the two other components of the total error must be imposed (see [5]):

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

2.2. Types of Problems with Regard to Their Computational Complexity and Estimates of Characteristics

The possibility to construct solutions to problems under conditions (1)–(3) can significantly depend on the type of the estimates that are employed.

It is well known [4, 5] that there are a priori and a posteriori, majorant and asymptotic, deterministic and stochastic estimates. Each type has its advantages and disadvantages. For example, a priori majorant deterministic estimates are guaranteed estimates, their calculation does not require the problem to be solved, and they are easy to calculate. On the other hand, such estimates are often too high and poorly suited for the quantitative analysis. The situation remains the same even if such an estimate is sharp because the problem on which the estimate is attained can be not typical for the class of problems and the capabilities of a computational algorithm remain unused on the other problems in this class.

Asymptotic a posteriori estimates can be close to the quantity being estimated, but this closeness is attained at the values of the parameter (which is changing) belonging to a certain domain of asymptotics, which is not always convenient for practical applications. Moreover, the calculation of such estimates is based on a solution of the problem, and its construction requires a considerable amount of computations.

The utility of using estimates of a certain type depends on the situation.

Let us describe the groups of problems in which it is suitable (under conditions (1)–(3)) to use asymptotic a posteriori estimates as the most accurate ones even though their computational complexity is high. These problems are as follows:

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

—large series of problems of the same type (the parameters vary in a small range, e.g., a functional minimization problem in which the phase space consists of solutions of a system of differential or integral equations);

—problems that are solved using CA-programs (see [9]) from problem-oriented libraries (to use these programs efficiently, it is desirable to know the exact values of the characteristics for the typical representatives of the class of problems);

—problems requiring a large amount of computations and that must be solved in a practically acceptable amount of time;

—problems that require highly accurate computations.

3. ON THE COMPUTATION OPTIMIZATION OPPORTUNITIES

The quality characteristics of solutions and computational procedures can be improved due to the use of various opportunities (see [3]). Here are some of them.

3.1. Opportunities for Decreasing Errors of Approximate Solutions

1. Due to refining the initial data ($\rho(E_H)$):

—refining the class of problems;

—correcting the initial data;

—detecting and refining the a priori information about the problem;

—improving the accuracy of the initial data;

—narrowing the class of problems by using as much information about the problem as possible.

2. Due to the solution method ($\rho(E_\mu)$):

—the use of optimal or almost optimal (in terms of accuracy) algorithms;

—optimization of the set of functionals that carry information about the problem (for example, optimization of the grid of nodes in numerical integration);

—increasing the number of functionals in the set;

—modification of the class of functionals (the use of a better optimal order of computational algorithms than in the original set of functionals);

—the complete use of the initial data to narrow the class of problems.

3. Due to rounding ($\rho(E_\tau)$):

—the use of optimal (in terms of order of accuracy) methods for highly accurate computations;

—narrowing the class of problems (in order to reduce the number of functionals in the set);

—the use of computational schemes that minimize the accumulation of rounding errors;

—improving the optimal order of accuracy by selecting a better class of functionals;

—increasing the computer word size;

—selection and modeling the rounding rules;

—the use of the optimized set of functionals.

3.2 Opportunities for Reducing the CPU Time Required to Solve the Problem

—refining the class of problems;

—the use optimal methods in terms of the execution time;

—the use of improved initial data (as a result, the use of less stringent constraints on the error estimates of the method and estimates of the rounding errors);

—improving the quality of error estimates of the method and estimates of the rounding errors;

—improving the accuracy of the parameter computation in the CP;

—matching the computational algorithm with the computer architecture;

—fast arithmetic (see [10, 11]);

—parallel computing;

—development of dedicated processors (the choice of a computer architecture that better suits the computational algorithm used to solve the problems in the class under consideration).

4. STEP-BY-STEP SCHEME OF THE COMPUTER TECHNOLOGY

In order to ensure that the resulting solution of the problem has the prescribed quality characteristics (1)–(3), one must know what is to be done, how this must be done, and in what succession.

The concept of the proposed technology is as follows.

1. Based on the conditions under which the desired solution of the applied problem (which is stated in the framework of a mathematical model) is to be used, requirements for the acceptable error of the approximate solution of the corresponding computational mathematics problem, requirements for the computation time, requirements for some other characteristics of the CP (e.g., computer memory), and requirements for the interpretation of the resulting solution (with regard to the estimates of the characteristics of this solution and of the CP) are formulated.

A CA-program is selected (or developed) that can solve the corresponding computational mathematics problem with the prescribed quality characteristics; it may happen that no such CA-program exists or it may be found that such a CA-program cannot be developed.[2]

2. The selected or developed CA-program is used to solve the applied problem with the prescribed quality characteristics on the selected computer.

The need for solving a problem with prescribed quality characteristics ($\rho(E)$, T, M) of the ε-solution and for deciding whether such a solution can be produced by the corresponding CA-program arises in the following cases:

—the problem must be solved up to a prescribed (or guaranteed) accuracy subject to certain constraints on the computation time and computer memory;

—the quality of the resulting ε-solution must be assessed in terms of its accuracy and computation time;

—even before solving the problem, it is necessary to know whether it can be solved with the prescribed quality characteristics ($\rho(E)$, T, M) to select (or develop) a CA-program to be included in a problem-oriented package of programs.

[2] In what follows, we assume that a computational algorithm and the corresponding program are the same things up to the rounding error.

The procedure for solving problems under conditions (1)–(3) can be considered as a sequence of steps or technology elements (see [1, 2]).

Below, we describe the sequence of steps (which may be used directly or modified) for solving problems with prescribed quality characteristics.

Step 1. Based on a systematic approach, the applied problem is stated in terms of the application field, and requirements for the desired ε-solution are formulated.

Step 2. A mathematical model of the applied problem is selected from the available models or is constructed; this model is used to describe the quantitative relationships between the elements of the model of the system (a process or phenomenon) and to find an ε-solution of the applied problem (in particular, it is used for carrying out a computational experiment, see [12, 13]).

Step 3. The mathematical model of the applied problem $P(I)$ with regard to its initial data belongs to a class K of computational and applied mathematics problems. The information $I = I(I_0, I_n(P))$ about the problem statement, about the desired ε-solution (the initial data, estimates of errors with which this data is specified or calculated), etc. is analyzed (see [5, 14–23]).

Later (when Step 3 is repeated) the problem $P(I)$ can be considered in a certain subclass $\tilde{K}$ ($\tilde{K} \subset K$) in order to extend the possibilities to satisfy conditions (1)–(3), or it can be reformulated and considered in another class of problems $\hat{K}$ (e.g., for ill-posed problems, see [24, 25]).

Step 4. The given requirements (1)–(3) for the values of the quality characteristics of the ε-solution are analyzed in view of the practical considerations (operational requirements of the process or phenomenon being modeled) and the system of units used in the mathematical model (typically, SI). This is done upon the problem of interest is scaled, if needed.

Step 5. An estimate of the error ($\rho(E_H(\cdot))$) of the unknown ε-solution is found (or improved) with regard to the uncertainty of the initial data). For that purpose, a computer model of computations, the initial data of the problem, and estimates of the error in this data are used. Available estimates of $\rho(E_H(\cdot))$ for problems of the corresponding class may also be used (e.g., see [26, 27]).

Step 6. For the estimate of $\rho(E_H(\cdot))$ thus found, the condition

$$\rho(E_H(\cdot)) < \alpha_1\varepsilon, \quad 0 < \alpha_1 < 1, \quad \text{e.g.,} \quad \alpha_1 = 1/3 \tag{8}$$

is verified. If (8) is fulfilled, then go to Step 8; otherwise, go to Step 7.

Step 7. To guarantee that the problem is solved up to accuracy (1), the requirement for the accuracy of the desired ε-solution should be (if possible) modified (e.g., $\varepsilon > 0$ can be increased) or $\rho(E_H(\cdot))$ should be improved. Go to Steps 4 or 5, respectively.

If $\rho(E_H(\cdot))$ exceeds the prescribed $\tilde{\varepsilon} > 0$ and cannot be decreased (if the problem is ill posed), the statement of the problem $P(I) \in K$ must be changed (go to Step 3) or the search for the ε-solution should be abandoned because it cannot be calculated within the adopted CM.

Step 8. Analyze the computational situation for solving problems of the class K (or the subclass $\tilde{K} \subset K$):

(a) No method is available for solving problems in K (or in $\tilde{K} \subset K$); hence, no algorithms for such problems are available and no theoretical estimates of their characteristics are known. Then, go to Step 9.

(b) Methods (or one method) for solving problems in K (or in $\tilde{K} \subset K$) are known, but there are no ready-to-use algorithms and CA-programs for the computer $c(Y)$ and no theoretical estimates of their characteristics ($\rho(E)$, T, M) are available. Then, go to Step 10.

(c) There are ready-to-use CA-programs for solving problems in K (or in $\tilde{K} \subset K$) in an available computer algebra system (such as Mathematica, MatLab, MathCAD, or Python), but no estimates of characteristics are available for deciding whether or not those CA-programs can be used to solve the problem $P(I) \in K$ (or $P(I) \in \tilde{K} \subset K$) with the prescribed quality characteristics (1)–(3). Then go to Step 16.

(d) There is a package of programs for solving problems in K that includes a CA-program for constructing an ε-solution of problems in the subclass $\tilde{K} \subset K$ such that provides estimates of the characteristics $\rho(E)$, T, and M simultaneously with the construction of an ε-solution of the problem $P(I) \in \tilde{K}$ with a given (in a certain range $D(\varepsilon)$) error $\varepsilon \in D(\varepsilon)$. Go to Step 17; otherwise, go to Step 10.

Step 9. For the class K (or the subclass $\tilde{K} \subset K$) that contains the applied problem $P(I)$, no solution method is available; therefore, it must be developed. In the development of such a method, in its justification and analysis of its characteristics (in the development of the theory and in proving the correspond-

ing theorems), known or new approaches to the solution of problems in K (or in the subclass $\tilde{K} \subset K$) can be used. Upon such a method has been developed, go to Step 10.

Step 10. There are known methods for solving problems in the class K (or in the subclass $\tilde{K} \subset K$) or such methods were developed at Step 9.

From the set of available methods for solving problems in the class K or in its subclass $\tilde{K}$, the "best" method in terms of the accuracy and performance is selected using the known a priori estimates of the quality characteristics (e.g., bounds on the error, estimates of the convergence, information and combinatorial complexity). To make this choice, the knowledge available in computer databases and the literature is used (see [5, 14–27]).

Step 11. Parameters of the computer that will be used to solve the problem $P(I) \in K$ (or $P(I) \in \tilde{K} \subset K$) are chosen based on the preliminary analysis of its complexity and requirements for the quality characteristics of the ε-solution; that is, the definition of the computer model $c(Y) \subset C(Y)$ is completed by selecting the parameters Y—the number of processors ($k = 1$ or $k > 1$), their type, the word size, the precision of arithmetic operations, estimates of the instruction execution time, computation modes, and constraints M_0 on the random access memory.

Step 12. On the basis of the method chosen at Step 10, a T-efficient algorithm for finding the ε-solution of the problem $P(I) \in K$ (or $P(I) \in \tilde{K} \subset K$) is developed and estimates of its characteristics are found using the following scheme.

(a) On the bases of the selected method, an algorithm $a \in A$ is developed for the computer $c(Y)$, which can be described in more or less detail, including a step-by-step description in a pseudocode. When Step 12a is repeated, a new algorithm may be developed or the initial algorithm $a \in A$ may be improved.

(b) An a priori estimate $M(\varepsilon, I, X, Y)$ of the computer memory that will be used by the algorithm $a \in A$ for constructing an approximate solution of the problem $P(I)$ is found or an available estimate is improved.

(c) Condition (3) is checked. If it is fulfilled, then go to Step 12d; otherwise, go to Step 11 (or 12, if the algorithm is fixed). This is done in order to ensure the fulfillment of condition (3) due to a modification of the initial or the development of another algorithm $a \in A$ (possibly, with the choice of a corresponding computer $c(Y)$).

(d) A priori estimates are found or improved. These are the estimate of the total error of the approximate solution $\rho(E(I, X, Y))$ and its three components $\rho(E_H(I, X, Y))$, $\rho(E_\mu(I, X, Y))$, and $\rho(E_\tau(I, X, Y))$ that occur, respectively, due to the inaccuracy of the initial data, the method (algorithm), rounding in the use of the algorithm $a \in A$, and the computer model $c(Y)$.

(e) Based on the estimate $\rho(E(I, X, Y))$, condition (1) is checked; that is, it is checked whether an ε-solution of the problem $P(I)$ can be calculated using the given information I_n. If no ε-solution can be found, the causes are detected, and an additional analysis beginning from Step 10 or 12 of the present scheme is carried out. If the chosen algorithm can find an ε-solution of the problem $P(I) \in K$ (or $P(I) \in \tilde{K} \subset K$), then go to Step 12f; otherwise, stop the search for the algorithm $a \in A$ and indicate why the solution of the problem cannot be ensured under condition (1).

(f) An a priori estimate $T(\varepsilon, I, X, Y)$ of the CPU time for the algorithm $a \in A$ is found or improved.

(g) Using the estimate $T(\varepsilon, I, X, Y)$, find out if the selected algorithm guarantees that an ε-solution of the problem $P(I)$ can be obtained under condition (2). If this is the case, then we conclude that the selected algorithm a belongs to the set $A(\varepsilon, T_0)$ ($A(\varepsilon, T_0) \in A(\varepsilon)$) of efficient (in terms of performance) (T-efficient) algorithms, which depends on T_0. If a is not T-efficient, the possibilities of ensuring its T-efficiency by modifying condition (2) and the construction of a T-efficient algorithm $a \in A$ by using optimization opportunities (Steps 12a or 12f) should be analyzed.

Remark. If the scheme described above cannot produce an algorithm $a \in A(\varepsilon, T_0)$ for finding an ε-solution of the problem, it is important to know exact (or close to them) lower bounds on the error of the approximate solution and estimates of the computational complexity of the problem. Using these estimates, one can conclude whether a solution of the problem with the prescribed quality characteristics can be constructed or this is impossible and the applied problem should be reduced to another class of computational mathematics problems or a computer of another class should be used.

If a T-efficient algorithm ensuring the prescribed quality characteristics is constructed, then go to Step 13; otherwise, go to Step 12.

Step 13. The T-efficient algorithm for solving the problem $P(I) \in K$ (or $P(I) \in \tilde{K} \subset K$) is implemented in a certain programming language for the selected computer $c(Y)$.

The CA-program for finding a solution of the problem $P(I)$ with the prescribed quality characteristics (1)–(3) must belong to one of the following classes of CA-programs.

1. The CA-program calculates an ε-solution, and the values of the control parameters X that ensure prescribed accuracy (1) from a certain range of accuracies $D(\varepsilon)$ ($\varepsilon \in D(\varepsilon)$) are calculated by the same program using a priori error estimates; the components of the vector X of control parameters can include, e.g., the number of iteration steps, grid size, etc.

2. The CA-program calculates an ε-solution and an a posteriori error estimate of the resulting solution. The construction of an ε-solution with the prescribed error $\varepsilon \in D(\varepsilon)$) is ensured by the corresponding selection of the control parameters X using the a posteriori estimate.

3. The CA-program calculates an approximate solution of the problem $P(I)$ but does not produce any error estimate. The prescribed accuracy $\varepsilon \in D(\varepsilon)$ is ensured by the user due to an appropriate choice of the control parameters X of the CA-program using the error estimates obtained by testing.

Step 14. The developed CA-program and the estimates of the quality characteristics $\rho(E)$ and T are tested using an appropriate test suite in accordance with the technology developed for testing the quality characteristics of CA-programs (see [9]).

If the testing results confirm that the developed or selected CA-program can solve the problem $P(I) \in K$ (or $P(I) \in \tilde{K} \subset K$) with prescribed quality characteristics (1)–(3), then go to Step 18; otherwise, go to Step 15.

Step 15. The CA-program is analyzed to find opportunities to improve it in terms of the characteristics that do not satisfy the requirements. The CA-program is optimized, and Step 12 is executed for the resulting modification. If all the optimization opportunities have bee tried but the quality characteristics do not satisfy conditions (1)–(3), then go to Step 11 or 12.

Step 16. If a CA-program for solving problems in the class K or its subclass $\tilde{K} \subset K$ was selected from an available library, it is tested (if needed) to determine the characteristics $\rho(E)$ (the accuracy of the solution) and T (computation time) in accordance with the testing technology described in [9].

If the selected CA-program ensures the solution of the problem $P(I) \in K$ (or $P(I) \in \tilde{K} \subset K$) with the prescribed quality characteristics in terms of accuracy and performance, then go to Step 18; otherwise, go to Step 8b or 8c.

Step 17. Using the estimates of $\rho(E)$, T, and M provided in the library for the selected CA-program, check if this CA-program can ensure the construction of an ε-solution of problems in the subclass $P(I) \in \tilde{K}$ up to required accuracy (1) $\varepsilon \in D(\varepsilon)$ under restriction (2) on the computation time. If the CA-program ensures the construction of an ε-solution of the problem $P(I) \in \tilde{K}$ satisfying conditions (1)–(3), then go to Step 18; otherwise, go to Step 8.

Step 18. Construct an ε-solution of the problem $P(I) \in K$ (or $P(I) \in \tilde{K} \subset K$) with prescribed quality characteristics (1)–(3) using the selected or developed CA-program.

Step 19. Interpret the computation results. If these results are inappropriate in a certain sense, go to Step 1 or Step 2.

CONCLUSIONS

The proposed computer technology for solving applied and computational mathematics problems with prescribed quality characteristics (1)–(3) as applied to specific problems (or classes of problems) provides the following capabilities.

1. Carry out a systematic approach to the problem statement and to its solution.

2. Analyze the computational situation (see Step 8) for solving the applied problem in the class K (in the subclass $\tilde{K} \subset K$) and identify the steps to be carried out to solve the problem with prescribed quality characteristics (1)–(3).

3. Based on the computational situation, develop a corresponding CA-program for solving the problem.

4. If possible, use a program from an available computer algebra system (such as Mathematica, MatLab, MathCAD, or Python) and select a program (if a handful of such programs are available) that provides the prescribed quality characteristics.

5. Use the proposed technology for solving applied problems with high requirements for quality characteristics (1)–(3).

REFERENCES

1. I. V. Sergienko, V. K. Zadiraka, M. D. Babich, et al., "Computer Technologies for Solving Applied and Computational Mathematics Problems with Prescribed Quality Characteristics," Kibern. Sistemn. Anal., No. 5, 33–41 (2006).
2. V. K. Zadiraka, M. D. Babich, A. I. Berezovskii, P. M. Besarab, and V. A. Lyudvichenko, "Algorithmic and Computer Tools for Ensuring a Desired Quality of Solving Applied Mathematics Problems," Upr. Sist. Mash., No. 5, 3–12 (2008).
3. I. V. Sergienko, V. K. Zadiraka, M. D. Babich, P. M. Besarab, and V. A. Lyudvichenko, "Opportunities for Optimization of Computations in Computer Technologies Designed for Solving Problems with Prescribed Quality Characteristics," in *Proc. of the Int. Symposium on Optimization of Computations*, Katsiveli, 2007, pp. 258–260.
4. V. V. Ivanov, M. D. Babich, A. I. Berezovskii, P. M. Besarab, V. K. Zadiraka, and V. A. Lyudvichenko, Characteristics of Problems, Algorithms, and Computers in Computational Mathematics Libraries, Preprint No. 84-36, Glushkov Institute of Cypernetics, Academy of Science of Ukraine (Kiev, 1984).
5. V. V. Ivanov, *Numerical Methods for Computers: Handbook* (Naukova Dumka, Kiev, 1986) [in Russian].
6. V. S. Mikhalevich, I. V. Sergienko, V. K. Zadiraka, and M. D. Babich, "On the Optimization of Computations," Kibern. Sistemn. Anal., No. 2, 65–94 (1994).
7. M. D. Babich, A. I. Berezovskii, P. N. Besarab, V. K. Zadiraka, and V. A. Lyudvichenko, "*T*-Efficient Algorithms for Computing ε-Solutions of Applied and Computational Mathematics Problems," Kibern. Sistemn. Anal., Part II, No. 2, 49–70, Part III, No. 3, 97–18 (2001).
8. J. F. Traub and H. Wözńiakowski, *A General Theory of Optimal Algorithms* (Academic, New York, 1980; Mir, Moscow, 1983).
9. M. D. Babich, V. K. Zadiraka, and I. V. Sergienko, "A Computing Experiment in the Problem of Optimization of Computations," Kibern. Sistemn. Anal., Part I, No. 1, 51–63, Part II, No. 2, 59–79 (1999).
10. M. A. Kartsev, *Computer Arithmetic* (Nauka, Moscow, 1969) [in Russian].
11. V. K. Zadiraka and O. S. Oleksyuk, *Computer Arithmetic of High-Precision Numbers* (Naukove vydannya, Kiev, 2003).
12. P. S. Krasnoshchekov and A. A. Petrov, *Principles of Model Construction* (FAZIS, Moscow, 2000) [in Russian].
13. A. A. Samarskii and A. P. Mikhailov, *Mathematical Modeling: Ideas, Methods, and Examples* (Nauka, Fizmatlit, Moscow, 1997) [in Russian].
14. A. A. Dorodnicyn, *Computational Mathematics. Encyclopedia of Cybernetics* (URE, Kiev, 1973), Vol. 1 [in Russian].
15. A. A. Dorodnicyn, M. F. Kaspshitskaya, and I. V. Sergienko, "An Approach to Formalization of Classification," Kibernetika, No. 6, 132–140 (1976).
16. G. I. Marchuk and V. I. Lebedev, *Numerical Methods in Neutron Transport Theory* (Atomizdat, Moscow, 1971) [in Russian].
17. N. S. Bakhvalov, N. P. Zhidkov, and G. M. Kobel'kov, *Numeical Methods* (Laboratoriya bazovykh znanii, Moscow, 2002) [in Russian].
18. V. V. Voevodin, *Mathematical Models and Methods in Parallel Processes* (Nauka, Moscow, 1986) [in Russian].
19. A. A. Samarskii and A. V. Gulin, *Numeical Methods* (Nauka, Moscow, 1989) [in Russian].
20. Yu. G. Evtushenko, *Methods for Solving Extremum Value Problems with Applications to Optimization Systems* (Nauka, Moscow, 1982) [in Russian].
21. F. P. Vasil'ev, *Numerical Methods for Solving Extremum Value Problems* (Nauka, Moscow, 1988) [in Russian].
22. I. N. Molchanov, *Computer Methods for Applied Mathematics Problems: Algebra, Approximation of Functions, Ordinary Differential Equations* (Naukova Dumka, Kiev, 2007) [in Russian].
23. J. Mathews and K. Fink, *Numerical Methods Using MatLab* (Prentice Hall, Upper Saddle River, N.J., 1999; Vil'yams, Moscow, 2001).
24. A. N. Tikhonov and V. Ya., *Solutions of Ill-Posed Problems* (Nauka, Moscow, 1986; Winston, Washington, 1977).
25. V. A. Morozov, *Regular Methods for Solving Ill-Posed Problems* (Nauka, Moscow, 1987) [in Russian].
26. M. D. Babich and V. V. Ivanov, "Investigation of the Total Error in Constrained Functional Minimization Problems," Ukr. Metrologich. Zh. **21** (1), 3–14 (1969).
27. V. K. Zadiraka, *The Theory of Fourier Transform Computation* (Naukova Dumka, Kiev, 1983) [in Russian].