DOI: 10.17725/rensit.2022.14.331

Computational experiment – nondimensionalization of equations, computational stability and program testing

Mikhail G. Evtikhov

Kotelnikov Institute of Radioengineering and Electronics of RAS, Fryazinsky branch, http://fire.relarn.ru/
Fryazino 141190, Moscow region, Russian Federation
E-mail: emg20022002@mail.ru

Vladimir G. Evtikhov

Moscow Polytechnic University, https://mospolytech.ru/
Moscow 107023, Russian Federation
E-mail: evg2002@yandex.ru

Received 17 June 2022, peer-reviewed 24 Juni 2022, accepted 01 July 2022

Abstract: **From the stages of the computational experiment, the stage of non-dimensionalization of the initial equation (system of equations) of the problem is considered - the replacement of its variables by the product of the corresponding dimensionless quantities by their units of measurement with subsequent transformations. Such a transition from a physical model to a mathematical (dimensionless) one makes it possible to obtain software implementations for research. A critical evaluation of its complexity is carried out and possible errors in the results are evaluated. At the same time, new versions of software are formed. Object-oriented programming tools and version control systems (for example, git) allow you to create versions of software tools adapted to different conditions of their use and for different types of users. Parallelization of work on versions is carried out. At the same time, for further software implementation, the set-theoretic language of formulas with partially recursive functions is effective. To implement versions with large amounts of calculations and data, high-performance computing systems based on software and hardware acceleration, parallel information processing and cloud architectures are used. As a rule, a difference model of the problem and iterative methods for solving it are constructed for a program version. Computational stability conditions are usually stipulated in modern instructions for standard program libraries. For new algorithms, it is necessary to analyze the stability of difference schemes based on the refinement of their spectral properties and the use of functional analysis methods. For storage and subsequent application of the results of computational experiments, it is advisable to use modern databases. As a kind of computational experiment, testing of alpha and beta versions of programs and their releases is also considered.**

Keywords: **computational experiment, problem statement, nondimensionalization, partial recursive function, numerical stability, computer literacy, tester, software development, software versioning**

UDC 167.2, 004.004, 004.007

For citation: Mikhail G. Evtikhov, Vladimir G. Evtikhov. Computational experiment – nondimensionalization of equations, computational stability and program testing. *RENSIT: Radioelectronics. Nanosystems. Information Technologies,* 2022, 14(3):331-340. DOI: 10.17725/rensit.2022.14.331.

Contents

1. INTRODUCTION

A computational experiment is an experiment conducted not on the original real object, but on a mathematical (information, simulation) model of the object using computational and logical procedures implemented by appropriate software tools on computer systems. Closures to the computational experiment concepts are numerical, mathematical and simulation experiments on a computer [1].

In [2], various approaches to the formation of mathematical models based on scientific laws and principles are considered. These approaches can be considered as ways of setting up computational experiments.

A special view of the computational experiment, closely related to the theory of probability, is being developed, for example, within the framework of the "planning of experiments" direction [3]. When designing new types of computers in the 70s, the possibilities of replacing some physical parts of a computer system with its software models were analyzed [4]. This efficient approach has now found its way into programming methods in the form of separating interfaces from their implementations (eg *h*-files and *cpp*-files in C++). The development of computational experiment methods and its central part, mathematical modeling, was accompanied by the clarification of a number of fundamental features of computational methods. In [5], a profound idea was expressed that programs for computers from a mathematical point of view are described by partially recursive functions. With strict definitions of algorithms, such a statement is considered as a hypothesis and is called "Church's thesis". From the point of view of an applied programmer, we mean mathematical constructions that are more widely understood than formulas based on radicals that are familiar to many researchers. For example, numerically solving an equation of degree 10 is a standard problem, but there is no general solution in radicals for an equation of degree 5. This does not mean that formulas can be dispensed with. In the formulas of the mathematics of partially recursive functions, the language of set theory and recursion, substitution, and enumeration of elements of sets are more widely used. For example, in [6], to description of fractals the L-system is used, i.e. recursive expression for sets: if $Z^{(0)}$ is a unit black square in the complex plane, then

$$Z^{(n+1)} = Z^{(n)} \cup \left(iZ^{(n)} + (1+i)2^n\right) \cup \left(-iZ^{(n)} + (1+i)2^n\right),$$

where $\cup$ is the operation of union of sets.

Generalizations of recursive formulas of set theory adequate to the subject of research, when necessary accurately and concisely record the results that already received and researched. However, it is difficult to find errors in these mathematical constructions, make changes and check them. Accordingly, debugging takes more than 90% of the development time of a new program and continues throughout its entire life cycle.

When writing already obtained theoretical solutions to problems, the language of formulas with recursive functions is effective for their further software implementation. This approach not only simplifies programming, but also allows professional mathematicians and designers to take part in solving complex problems of algorithm optimization and information storage.

A significant difficulty of the computational experiment is the computational instability [7]. The instructions for the standard libraries of programs (classes) usually stipulate the conditions for computational stability and other

conditions investigated for the correct use of software tools. The use of libraries for general sparse matrices [8] expands the possibilities for developing new numerical methods.

There are developed software tools and software libraries for solving various problems of modeling and computational experiment. These tools are deeply thought out and researched. It would seem that there is opportunity to further develop physics, other sciences and technologies. At the same time, the stock of achievements in the field of functional analysis is far from exhausted. A solid groundwork in the theory of probability [9,10] is the basis for many successful applications of the computational experiment. Obviously, programming cannot develop without experimental research also in the field of programming itself.

In recent years, outsourcing (attraction of external specialists) in the field of software product testing has become inevitable. The work of a tester can be interpreted as an experimental study of a software product; this activity determines the quality and competitiveness of software products. A few years ago, no special skills or education were required from a tester. Currently, special training is already required. Simulation and computational experiment become relevant at all levels of modern programming. Related to testing interesting and practically valuable ideas are discussed in many Internet publications. A general view, a "pyramid of tests", is given, for example, in [11].

The purpose of this article is to reveal the concept of a computational experiment, using the example of a simple problem, taking into account the point of view of applied programmers: the role of nondimensionalization of equations, the computational stability of numerical methods, and the development of programs by versions.

2. STAGES OF THE COMPUTATIONAL EXPERIMENT

It seems appropriate to state some topical problems of a computational experiment using the example of solving a relatively simple problem.

2.1. Problem statement

For example: on a steep seashore, a cannon fires at a given angle to the horizon. How far will the projectile fly if you can ignore air resistance and assume the Earth is flat.

The flight of the projectile must satisfy Newton's second law:

$$m\frac{d^2y}{dt^2} = -mg; \quad m\frac{d^2x}{dt^2} = 0,$$

m is the mass of the projectile, x is the horizontal coordinate of the projectile in flight, y is the vertical coordinate, t is the time, g is the acceleration of gravity near the surface of the Earth.

Note that the mass can be reduced. Therefore, the solution will be independent of the mass of the projectile. If there were also additional terms describing air resistance in the equations, then they would be divided on mass. The larger the mass, the less their influence would be. We temporarily neglect air resistance, so the equations are reasonable and adapted to the massive and small bodies.

V_0 is the speed of the projectile at the moment of its departure from the cannon, α is the angle at which the cannon fires, h is the height at which the cannon stands.

2.2. Nondimensionalization of equations

The non-dimensionalization of equations is a subtle, not obvious to everyone, and, perhaps, the most important stage of the study. This is sometimes underestimated and leads to serious and even catastrophic consequences. This stage

requires consultations with specialists involved in full-scale experiments.

The laws of the natural sciences are usually written in the SI system of units. The advantage of the SI system of units is its linkage to the scale of ordinary everyday experience.

We write the physical variables as the product of a dimensionless quantity (let's mark it with a wave) and its units of measurement (in square brackets):

$$x = \tilde{x}[\text{m}];\ y = \tilde{y}[\text{m}];\ h = \tilde{h}[\text{m}];\ t = \tilde{t}[\text{s}]$$

$$g = \tilde{g}[m \cdot s^{-2}],\ \tilde{g} = 9{,}8;\ V_0 = \tilde{V}_0[m \cdot s^{-1}]$$

Here all lengths are measured in meters, time is in seconds, g is the acceleration of gravity near the Earth's surface, speed is measured in meters per second.

The standard library functions assume that the angles are given in radians, i.e. α is already a dimensionless quantity. The angle given in degrees is nondimensionalized using the relation $\alpha = \frac{\pi\tilde{\alpha}[\text{deg}]}{180[\text{deg}]}$, π is the ratio of the circumference to the diameter, $\tilde{\alpha}$ is the angle in degrees.

The procedure for dimensionless equations is that the variables, together with their dimensions, are substituted into the original equations.

It turns out that the dimensions can be reduced. Moreover, all dimensions must be canceled out. After non-dimensionalization, only dimensionless variables should remain in the equations:

$$\tilde{y} = -\frac{\tilde{g}\tilde{t}^2}{2} + \tilde{t}\tilde{V}_0 \sin\alpha + \tilde{h},$$

$$\tilde{x} = \tilde{t}\tilde{V}_0 \cos\alpha.$$

The form of the dimensionless equation must not have to exactly repeat the form of the original equation. If it turns out to be necessary to set, for example, the speed not in meters per second, but in kilometers per hour, then additional factors will appear in the non-dimensional equations. Sometimes artificial units of measurement are introduced specifically for this task.

When analyzing some models, the phisical dimensions of quantities not can be reduced. This case is result of serious misunderstanding of problem, either result of equations errors. Often with the help of non-dimensionalization, we can identify untenable theories claims.

2.3. Model programming

The nondimensionalization procedure is a transition from a physical model to a mathematical one. There are no grams, meters, or rubles in the computer's memory. Standard programs help to convert the dimensionless values of the model into computer bits and bytes. Only with the help of dimensionless mathematical model, programmers get the opportunity to write programs. In the case of a cannon problem, you should write functions that calculate $\tilde{x}$ and $\tilde{y}$ for a given $\tilde{t}$. A good manner would be to make the model as a class and put the acceleration $\tilde{g}$, height and angle in class fields, and $\tilde{x}$ and $\tilde{y}$ put calculation as class methods.

The mathematical model is useful precisely for the task for which it was developed. Its extended interpretations are possible, but doubtful. A simplified model can be a tool for solving more complex problems and a source of test cases for other more complex models. You should start with an extremely simplified model, and then develop versions of programs. Newton's idea of extremely simple experiments is especially relevant in a numerical experiment and is based on the possibility of using inherited classes.

2.4. Problem solving

The projectile flight time is the solution of the equation $\tilde{y} = 0$. The class method for obtaining this time can be implemented

completely by yourself, but it is better to use standard programs. It's not about whether it's easy or hard to solve a quadratic equation. Standard numerical methods and programs can numerically solve even such equations for which there are no solutions in the form of familiar formulas with radicals. After calling the methods of the library classes, in our case, it is necessary to provide for the selection of a solution in which $\tilde{t} > 0$. Having obtained the flight time of the projectile, we obtain a solution to the problem using the class method to calculate $\tilde{x}$. It remains to formalize the entire solution of the problem as another class method.

2.5. Evaluation of the problem solution. Cycling back to the level of thought experiments

After solving the problem, the turn comes to the most unpleasant surprise question: how to evaluate the error of this solution? Without an estimate of the error, physical quantities are meaningless. Estimates of errors in the initial parameters of the model correspond to the stages of a thought experiment and non-dimensionalization. The numerical model makes it possible to vary its parameters and estimate the result error. Sometimes this is easier than looking for derivatives of solution formulas.

At the stage of critical evaluation of the solution, ideas for the next version of the program arise. Work on the current version is still very far from completion, but it is already becoming possible and of fundamental importance to work on a new version. The technology of parallelization of work on versions appeared, apparently, in the aircraft industry, but it is possible even earlier. All modern programs are developed by versions. Existing version control systems allow you to do this quite conveniently.

2.6. Return to model programming

How to convert the solution for its storage and further use? Collecting, storing and processing big data is becoming a difficult problem. In [12], various types of modern databases and high-performance computing based on hardware-software acceleration, parallel information processing, and cloud architectures are discussed.

2.7. Return to the level of natural experiments

The data array obtained as a result of solving modeling problems and experiments must be stored in the database, and also brought to the consumer in a form understandable to him. Such tasks of developing database interfaces are becoming more and more relevant. Ability to develop web applications (including gadgets) has already become very desirable competence of a modern programmer [13]. It intersects with tasks computational experiment and has numerous directions. For example, wiki projects. They allow users to use the well-known Wikipedia interface. Some firms create their own specialized wikis independent of the main Wikipedia. They contain related materials for technically complex products, including software. It is possible to organize sophisticated automated searches on the main Wikipedia and other wikis. In [14], an example of a complex search, comparison and visualization of historical data is given. All these possibilities are promising for application in research and experimental work.

2.8. Model levels

As a result, the computational experiment can be represented as interconnected experiments with 4 layers of models:

- Thought experiment based on literature data, understanding how to evaluate model errors.

- Full-scale experiment with the development of interfaces.
- Mathematical model and experiments with databases.
- Solving mathematical problems, analysis of computational stability of algorithms using functional analysis tools.

3. COMPUTATIONAL INSTABILITY

Now in the previous problem we will have to take into account air resistance. In the new version we will have to build difference model, as well as an iterative method for its solutions. Iterative algorithms are efficient and simple, it is almost impossible to refuse them. Usually computational instability appears in iterative algorithms.

The essence of computational instability is the accumulation of computational errors. Computational instability is the main problem of numerical methods and experiments. The first attempts to use computational models of physical processes often turned out to be computationally unstable. They give results, that far from true solutions (for example, Richardson's scheme [7]).

The simplest model of oscillations with frequency ω can be described by a complex equation:

$$\frac{dX}{dt} = i\omega X;\ X(t=0) = A.$$

(for example, the Schrödinger equation at zero momentum in a homogeneous potential).

If A is a real number, then the solution to this equation will be oscillations with amplitude A, frequency ω and period $T = 2\pi/\omega$:

$$X = A\exp(i\omega t) = A\cos(\omega t) + iA\sin(\omega t),$$
$$\mathrm{Re}(X) = A\cos(\omega t);\ \mathrm{Im}(X) = A\sin(\omega t).$$

On the plane (Re(X), Im(X)) you should get a circle with radius A.

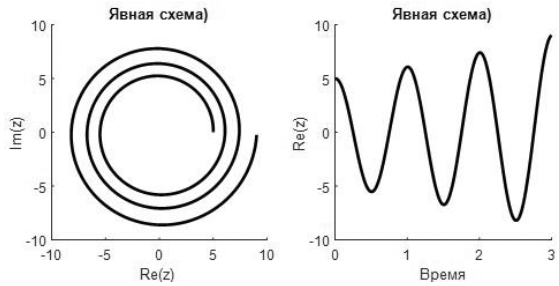

Fig. 1. *Explicit difference scheme.*

We choose a time step $\tau << T$ and approximate the derivative as

$$\frac{dX}{dt} \approx \frac{X_{i+1} - X_i}{\tau}.$$

we get

$$\frac{X_{i+1} - X_i}{\tau} = iwX_i \Rightarrow X_{i+1} = X_i(1 + iw\tau);\ X_1 = A.$$

Let, for simplicity, ω = 2π, then the oscillation period should be 1. Let τ = 0.01. Such an approximation of the equation seems natural, but this appearance is deceptive (**Fig. 1**).

The reason for this parasitic increase in amplitude is that small errors accumulate at each iteration step, and as a result, the solution deviates more and more from the true one. This phenomenon is called computational instability. As the step τ decreases, the parasitic increase in the amplitude decreases, but does not disappear.

A difference scheme, when the source value is taken at the current time layer, is called an explicit difference scheme. An implicit difference scheme is possible when the source function is taken not on the current, but on a new time layer (**Fig. 2**):

$$\frac{X_{i+1} - X_i}{\tau} = iwX_{i+1} \Rightarrow X_{i+1} = X_i / (1 - iw\tau);\ X_1 = A.$$

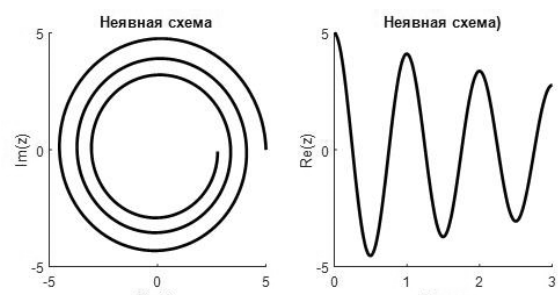

Fig. 2. *Implicit difference scheme.*

This difference scheme is considered to be more successful. But now there is parasitic attenuation of the amplitude. The amplitude is also not preserved, which is also far from always acceptable. The computational stability of the difference scheme does not guarantee the absence of such computational attenuation. In [15], one can get acquainted with the methods of spectral analysis of difference schemes.

The examples in Fig. 1 and Fig. 2 reveal the opposite of the concepts of "computational stability" and "robustness" (resistance to outliers). The preferred difference scheme for a computational experiment should be the least robust of the stable ones. The question arises: "Is it possible, after all, to calculate the oscillations so that the amplitude is preserved?". It is enough, for example, to use the Crank-Nicolson difference scheme [16] (**Fig. 3**):

$$\frac{X_{i+1}-X_{i-1}}{2\tau}=iwX_i \Rightarrow X_{i+1}=X_{i-1}+$$
$$+2iw\tau X_i;\ X_1=A;\ X_2=(1+iw\tau)A.$$

The effects of computational instability manifest themselves not only in oscillation models, but also when oscillations are not visible, for example, in Richardson's three-layer explicit difference scheme for the heat conduction problem [7].

Many problems of the computational experiment have long been solved, and the solutions have been repeatedly tested. In order to evaluate other people's non-trivial solutions, in the new version of the program it would be worthwhile to give more attention to the recommendations of the libraries of standard programs. Theorems on the properties of difference schemes [7] are quite general. In [17], for example, for the equations of mathematical physics, the possibility of combining difference schemes with a fast Fourier transform and embedding solutions in the Sobolev space was realized. Using this approach, it was possible to obtain by numerical methods not only the self-constructed of dissipative structures and magnetic domains, but also solitons.

In the case of nonlinear problems, it is difficult to prove computational stability, but additional checks can be included in the computational algorithms. The emergence of programs for working with sparse matrices [8] expands the possibilities for implementing numerical methods and makes the application of the basic definitions and approaches of the theory of difference schemes even more relevant [7]. The mathematical approach to the analysis and justification of the synthesis of difference schemes implies familiarity with functional analysis and the ability to immerse difference schemes in completed spaces [18].

Fig. 3. *Almost constant oscillation amplitude.*

4. VERSIONS OF SOFTWARE PRODUCTS

In the 1990s, a certain classification developed for software versions. Alpha version – the first versions of the developed program. They are workable, but only the developers themselves can really use them. Beta version is a more user-friendly version of the program. It is assumed that the user

uses it at his own peril and risk, that he himself will deal with his questions.

Initially, beta versions were intended for a limited circle of consumers qualified enough to use the program for free and without claims, despite possible errors and failures. Now a wider audience is participating in beta testing. The formation of this circle of people ensures further promotion of the product on the market.

GitHub cloud service allows programmers to share their developments and software. It supports materials quite conveniently. Releases – further improvement of the software product. The release numbering includes the version number and the revision number. For example, Windows 3.1 was the first widespread version of Windows. Sometimes versions are referred to by the year they were released, or they are given special names.

When developing and using software, most of the work time is spent on finding and fixing bugs, i.e. errors, inconsistencies with program specifications, inconsistencies with user expectations. These inconsistencies can be eliminated only in the simplest cases by formulating and testing hypotheses. Bug finding and testing problems are advisable to consider as a kind of computational experiment with version of program.

Modern browsers have modes that allow you to track document structures and related data, display formats, messages sent between form elements. Known software tools can record and reproduce the actions of tester during testing. Test results should be accumulated in a database with the ability to display in a form convenient for further use. The volume of required manual testing can and should be reduced by automating the synthesis of tests and the implementation of the testing process [11]. To perform many operations in parallel in the development and maintenance of software products, some firms use a outserting policy. There are firms that specializes in software product testing and new specializations of programmers.

If the elimination of a bug affects the work of other parts of the program, then would reasonable its postponing eliminating to a new version. Another simplified way to get rid of complex bugs is to create a specialized version with more narrow capabilities. Sometimes a demo version is useful. These bugs handling methods are to the levels of a mathematical model and a full-scale experiment, programming. But a level is deepest, at which programs are not explored, but are used as a means of solving problems.

5. CONCLUSION

The paper considers three aspects of a computational experiment: non-dimensionalization of the original model, computational instability of solutions, and testing as a kind of computational experiment. It is shown that at the stage of non-dimensionalization, the physical model is transformed into a mathematical one. This transformation makes it possible to reasonably proceed to the next stage of the computational experiment – programming. Parallelization of the development of algorithms and program versions for different users and purposes is based on an assessment of data complexity and errors. Object-oriented programming tools and version control systems are used.

For complex calculations and big data, it is advisable to develop software versions based on high-performance computing tools: on hardware-software acceleration, parallel information processing and cloud architectures.

As a rule, the new version of the program and a difference model are built to solve the problem by iterative methods. Conditions for computational stability usually specified in modern instructions for standard software libraries. For new algorithms, when analyzing stability of difference schemes their spectral properties are refined and methods of functional analysis are used.

For storage and subsequent use, the obtained solution data are converted for their accumulation in modern databases. If necessary, you can use distributed databases and cloud storage.

As a kind of computational experiment, testing and use of alpha and beta versions and releases are also considered.

REFERENCES

1. Mikhailov AP, Chetverushkin BN. COMPUTATIONAL EXPERIMENT. *Great Russian Encyclopedia,* vol. 6, p. 161. Moscow, 2006. Electronic version (2016), https://bigenc.ru/mathematics/text/2379725.
2. Samarskii AA, Mikhailov AP. *Principles of Mathematical Modeling. Ideas, Methods, Examples*. London and New York, Taylor and Francis Publ., 2002, 349 pp.
3. Krasovskii GI and Filaretov GF. *Planirovanie eksperimenta* (Design of Experiment), Minsk: Belarus State University Publ., 1982, 302 p.
4. Baytser B. *Architecture of computer systems,* v. 1. Moscow, Mir Publ., 1974, 498 p.
5. Kolmogorov AN(author), Shiryaev AN (editor). *Selected Works III: Information Theory and the Theory of Algorithms*. Astoria, NY 11106, Springer, 2012, 304 p.
6. Arzamastseva GV, Evtikhov MG, Lisovsky FV, Mansvetova EG. Light diffraction by fractals: comparison of experimental data with the obtained by numerical methods fourier images of the object pictures. *RENSIT: Radioelectronics. Nanosystems. Information technologies,* 2017, 9(2):221-228. DOI: 10.17725/rensit.2017.9.221.
7. Samarskii AA. *The Theory of Difference Schemes*. Boca Raton, CRC Press, 2001, 786 p.
8. Tewarson RP. *Sparse Matrices*. NY, Acad. Press, 1973.
9. Cramer H. *Mathematical Methods of Statistics*. Princeton University Publ., 1999, 575 p.
10. Kolmogorov AN. *Selected Works II: Probability Theory and Mathematical Statistics*. Astoria, NY 11106, Springer, 1992, 615 p.
11. Ham Vocke. *The Practical Test Pyramid*. [Electronic resource] // martinfowler.com. URL: https://martinfowler.com/articles/practical-test-pyramid.html (date of the application – 12.04.2022).
12. Evtikhov VG, Evtikhova NV, Evtikhov MG, Evtikhov MV. *Vysokoproizvoditelnye vychisleniya* [High-performance computing]. Kazan, Buk Publ., 2020, 150 p.
13. Evtikhov VG, Evtikhova NV. *WEB-razrabotka v obrazovatelnykh i vikipediynykh proektakh* [WEB-development in educational and Wikipedia projects]. Kazan, Buk Publ., 2018, 112 p.
14. Evtikhov MV, Evtikhov VG. Vizualizatsiya istoricheskikh dannykh [Visualization of historical data]. *Molodoy uchenyi*, 2019, 1(239):1-5 (in Russ.).
15. Godunov SK, Ryabenky VS. Raznostnye skhemy [*Difference schemes]*. Moscow, Nauka Publ., 1973, 440 p.
16. Marchuk GI. *Methods of Numerical Mathematics*. Berlin-Heidelberg-New York, Springer-Verlag, 1975, 316 p.

17. Evtikhov MG. On the numerical solution of the equations of mathematical physics with nonlinear sources. *Journal of Communications Technology and Electronics,* 2007, 52(8):1-4.
18. Anuchina NN, Babenko KI, Godunov SK, Dmitriev AN, Dmitrieva LV, D'yachenko VF, Zabrodin AV, Lokucievskij OV, Malinovskaya EV, Podlivaev IF, Prokopov GP, Sofronov ID, Fedorenko RP. *Teoreticheskie osnovy i postroenie chislennyh algoritmov zadach matematicheskoj fiziki* [Theoretical foundations and design of numerical algorithms for problems of mathematical physics]. Moscow, Nauka Publ., 1979, 295 p.