subdomains was chosen. Figure 3 shows the computed domain of the plane (ϕ, ξ). Here $\xi_0 = 0.1$, $\xi_1 = 2.4$, $\xi_2 = 3.6$. The rectangles 9, 18, 27, 36 were added to the computed domain, and by periodicity they are identical with the rectangles 1, 10, 19, 28.

In the pivotal condensation along the ξ–axis each subdomain of the partition included 4 of the numbered rectangles (for example, 21, 22, 30, 31 or 11, 12, 20, 21). The horizontal pivotal condensation along the ϕ–axis was taken in the domain 1–18. The whole of this domain was divided into three horizontal strips. Here only the two lower strips were used as the subdomain of the partition (for one iteration), or the two upper ones (for the other iteration). In these subdomains the convergence of the iterations was considerably slower than in the others. We therefore increased the number of iterations. Additional iterations were also performed in the region of the wake (in Fig. 3 these are the rectangles 19, 20, 28, 29, 37–40).

The mesh steps were chosen from an analysis of the norms of the matrices described in section 5. Depending on the domain and the direction of the pivotal condensation, and also to control the results the steps were varied within the following limits: $\Delta\phi = \pi/40$ or $\pi/80$; $\Delta\xi = 0.1, 0.05, 0.025$ or 0.0125.

The final results are shown in the graphs.

Figures 4 and 5 give the steady state values of the vortex on the boundary of the body ($\xi = \xi_0 = 0.1$ and close to it ($\xi = 0.3$ and $\xi = 0.5$) for Re = 20 and 40 respectively.

Figures 6 and 7 show the stream lines (ψ = const) in dimensionless coordinates (x, y) for Re = 20 and 40 respectively.

Translated by J. Berry

REFERENCES

1. KOCHIN, N.E., KIBEL, I.A. and ROZE, N.B. *Theoretical hydrodynamics* (Teoreticheskaya gidrodinamika), Part I. Fizmatgiz, Moscow, 1963.
2. DORODNITSYN, A.A. *Lectures on numerical methods of solving the equations of a viscous fluid* (Lektsii po chislennym metodam resheniya uravnenii vyazkoi zhidkosti), VTs Akad. Nauk SSSR, Moscow, 1969.

U.S.S.R. Comput. Maths Math. Phys. Vol. 18, pp. 149–155 0041–5553/78/0301–0149 $7.50/0

PRINCIPLES OF THE DEVELOPMENT OF APPLIED PROGRAM PACKAGES FOR PROBLEMS OF MATHEMATICAL PHYSICS*

V.Ya. KARPOV, D.A. KORYAGIN and A.A. SAMARSKII

Moscow

(*Received* 27 *September* 1977)

A REVIEW is given of problems facing the developers of applied program packages.

1. At the present time the level of development of computational technique, programming and numerical methods for problems of mathematical physics makes it possible to put on the agenda of the day the topic of the creation of mathematical models of complex physical and technical engineering processes and objects. The numerical investigation of such models, reflecting fairly fully the fundamental properties of the actual object permits us to understand more deeply the nature of the object and reduce the number of expensive physical experiments usually necessary at various stages of the realization of large scientific and technical programs.

We indicate as examples two large-scale problems of a complex nature whose solution requires a preliminary comprehensive theoretical investigation using computers.

A key problem of nuclear power is the design of a nuclear reactor, which is a complex technical installation. Its design and the control of its operation require the simultaneous solution of problems of

* *Zh. vȳchisl. Mat. mat. Fiz.*, **18**, 2, 459–467, 1978.

various branches of physics and engineering. These are problems of nuclear physics and nuclear theory, problems of neutron and radiation transfer, problems of heat transmission, gas-dynamics and of the theories of elasticity and strength.

Another large-scale scientific and technical problem is that of controlled thermonuclear synthesis. In the first place it requires the investigation of the behaviour of a plasma under different extremal conditions: at high temperatures and densities, at low densities, and in the presence of electromagnetic fields and thermonuclear reactions. The processes taking place in a plasma are essentially non-linear, and their theoretical investigation presents enormous difficulties. Several installations for obtaining thermonuclear synthesis exist. Comparison of them with each other and optimization by mathematical simulation must precede the construction of these large-scale installations.

Also of a complex nature are such important problems as the automation of the processing of the results of physical experiments, the solution of problems of chemical kinetics and the calculation of building constructions.

2. Computer applications to the solution of problems of a complex nature require the use of new, more modern forms of programming and the creation of highly productive software. This naturally involves the reconsideration and refining of some of the approaches and views established in programming, and also the determination of the fundamental current and prospective problems.

Beginning with the middle sixties two directions were fairly clearly distinguished in programming: applied programming, that is, the creation of algorithms and programs for solving problems in various fields of application of computers, and system programming, whose purpose is the development of various facilities of the program equipment of the computer. The activity of the teams representing these directions is characterized on the one hand by mutual collaboration, primarily expressed by the fact that the developers of applied programs use the "means of producing" programs created by the program-systemizers. This collaboration is based on the identity of the main purpose, namely the creation of a program availability permitting extensive possibilities for the efficient solution of problems by the computer. On the other hand, there is inherent in the operations carried out in both directions a certain separatism, following from the different nature of their problems and reflected both in the very methods of programming, and also in the nature of the organization of the work of the teams.

Over some years the negative aspects of this separatism had no serious effect on the efficiency of the developments of applied programming. This is explained by the fact that one of the important problems of system programming – the creation of facilities for the formal description of algorithms for solving problems, and apparatus for the translation of the formalized descriptions into computer language – was stated at the very beginning in connection with the need to automate the programming of applied problems. We note that the formulations of a whole series of other problems of systems programming were also drawn from the practice of applied programming. This practical involvement was due to the fact that, despite the separatism indicated, the facilities created by the system programmers were widely used by the applied programmers, and practically completely satisfied their systems enquiries. In those cases where the existing systems facilities were insufficient, they were developed by the applied programmers independently, such developments as a rule reducing to the modification (to some degree) of existing system facilities.

3. The confirmation of what is stated above can be seen by considering the evolution of programs for solving problems of mathematical physics. In the period of the formation of system programming the degree of complexity of a program was usually estimated only from the point of view of its size (the number of instructions). On average these were programs containing 10^3–10^4 instructions. For the processing of such programs it was sufficient to use some problem oriented language (ALGOL, FORTRAN) and a corresponding translator.

As the power of computers increased the applied programmers passed on to the creation of large programs (program complexes), each of which was intended for the solution of a complex, but specific, problem. Here not only the size of the programs was increased (10^4–10^5 instructions), but what is more important their logic was made essentially more complex. This stage of the evolution of applied programs was characterized by the emergence of a new approach to the development of them, called the modular principle of programming. This principle was primarily based on the possibility of representing and solving a complete problem as an aggregate of relatively independent physical subproblems. As a rule each physical subproblem is in turn represented as an aggregate of relatively independent mathematical problems. The use of such a structure was in fact the initial idea of modular programming. The transition to the modular programming principle was expressed by the fact that the solution of a problem is essentially secured not by one program but by a group of interconnected programs forming an applied program complex. These programs, specially designed so as to guarantee the possibility of their interconnection with various other programs, were given the title of moduli. In connection with the new approach to programming languages an apparatus of subprograms (procedures) was developed, serving as a basis for the realization of the modular programming principle, and special systems of facilities ensuring the static or dynamic formation of a complex of a certain structure were also developed.

In practical respects an important drawback of such programming complexes consisted of the fact that they were oriented only toward the solution of one and more rarely of several specific problems of a definite class. But the application of a complex of this kind for the solution of a problem, not foreseen in its design despite the modular structure of the complex, as a rule involved laborious debugging, in the final analysis commensurate with the work required to compile a completely new program. This deficiency was a consequence of both the imperfection of the methods of developing applied programs, and also of the scantiness of the possibilities presented by the facilities of the system.

Indeed, in the development of applied complexes the possible versions of the mathematical model were taken into account by including in the complex all the necessary moduli and by using in the process of its operation so-called controlling parameters. It is clear that with this approach the flexibility and universality of the complex was determined by the amount of foresight with which the moduli and controlling parameters were chosen. It must also be noted that from the point of view of each specific problem such a program complex is characterized by a certain redundancy, since it contains "superfluous" moduli and data fields, that is, moduli and data fields not used in the solution of this problem. So far as the scantiness of the possibilities of the system facilities is concerned, this was mainly expressed not so much by the absence of convenient facilities for modifying texts and the rearrangement of the structures of applied programs (in some form services of this kind were provided by developed computational systems), as by the absence of unified methods of using these facilities. In other words, the solution of questions of the coordinated use and organization of the interaction of various facilities, for example, such as the editor of texts, the translator and the data control system, were practically completely left to the applied programmers. We emphasize that it was precisely in connection with this fact that the applied programmers first seriously encountered the need to revise, modify and extend the possibilities of the existing system facilities.

4. We have already remarked that the present stage of development of programming is connected with the transition in computer applications to the solution of problems of a complex nature. In this connection we would like to mention three facts which in our view most precisely reflect the current position of affairs in programming and the interrelations between the two directions of this discipline:

a) the conduct of a computational experiment;

b) the creation of a stock of programs;

c) the universality of the system facilities.

The most important factor determining the nature of the operations in modern mathematical physics is fact a). In essence the method of the computational experiment is the fundamental theoretical method of investigating complex physical and technical engineering processes and objects. For physical problems the fundamental stages of the computational experiment consist of the following [1]:

1) the choice of the physical approximation and the formulation of the mathematical model (as a problem of mathematical physics);

2) the choice of the discrete model approximating the initial mathematical problem (for example, the construction of a difference scheme), and the development of a computational algorithm for solving the equations;

3) the creation of a program for realizing the computational algorithm;

4) calculations and analysis of information;

5) analysis of the results, comparison with the physical experiment, revision and improvement of the physical model, and if necessary, repetition of all the stages from the beginning.

These items 1) – 5) describe one cycle of a computational experiment. If the purpose of the study of some physical process is its optimization (that is, finding and creating conditions for which the process proceeds optimally with respect to some characteristic parameters), then the performance of the computational experiment requires the repetition of the given technological cycle until the result sought is attained. From the programming point of view a computational experiment is characterized by the fact that for every physical model for the purpose of establishing a connection between the physical experiment and the computational experiment it is necessary to solve a large number of versions (varying the defining parameters of the problem) and also to change (refine) the physical model itself. This feature (the "multivariable" and "multimodel" nature) of a computational experiment manifests itself in repeated changes in the program realizing the computational algorithm, these changes affecting both the structure of the program as a whole, and also the individual fragments of the program realization of the algorithm.

In fact the development of a program for a computational experiment develops into the creation of a large-scale programming system (with a volume of 10^5-10^6 instructions), characterized by a large number of components and a variety of their interactions. Obviously the only realistic way of providing a computational experiment with programming facilities permitting programs to be rapidly collected and the computational algorithm and physical model to be simulated, consists of the use of the modular programming principle combined with the development of system facilities. The fruitfulness of the modular principle of programming is ensured by the important fact that the description of the same physical processes occurs in different complex problems. Moreover, different physical processes are often described by the same equations. For example, the same equations describe such processes as diffusion, thermal conductivity, and magnetization. The difference between the processes is revealed only by the physical meaning of the coefficients of the equations and the unknown function. Consequently, in the optimal choice of a set of mathematical and physical moduli their number may be less than the number of physical problems solved with their aid.

In this connection it is appropriate to turn our attention to fact b), which is the result of previous efforts of the programmers and gives a basis for optimism. Up to the present time there has been accumulated in applied programming and is being supplemented at an increasing tempo, a considerable stock of programming aids, that is, various mathematical methods and algorithms for the solution of individual problems described in programming languages. In a number of cases these programming aids have been well checked, documented and are widely available. As an example we can cite the national state stock of algorithms and programs [2] and two international stocks of physical programs: the library of the Queen's University of Belfast (Northern Ireland) [3] and the library of reactor programs at Ispre (Italy). The existence of a fairly complete stock of programming aids enables us to pass to a qualitatively new level in applied programming. Now we can and must put on the agenda of the day the question of the creation of an extensive program service of individual classes of applied problems, or considering the question more widely, of individual sections of applied activity. This service must be so complete that the development of a particular program of a given class, characterized either by the large number of moduli used, or by their great complexity and the possibility of multiple variation, the variety of their interactions, or the aggregate of all these factors, would no longer be a unique problem, but would represent merely one of the working aspects of a specific applied activity.

It is quite obvious that this formulation of the question assumes the use of adequate system facilities, which we have already mentioned when speaking about the program provision of a computational experiment. And now we consider fact c), concerning the possibilities of existing system facilities from the point of view of modern applied programming. The fact is that the system facilities developed up to the present time, despite their practical bias, were as a rule oriented toward some "generalized" user. Of course, this assured them of an extremely wide application, but at the same time led to the position where from the point of view of each specific applied activity they were not entirely adequate. In other words, the possibilities realized by the system facilities were determined by taking into account general and not specific applied interests. The more the sphere of applied programming was extended and its methods perfected, the more it was found that the services guaranteed by the universal system facilities (for example, debugging facilities), were insufficient for the successful execution of the operations, and also, which is more characteristic, the use of these facilities for special purposes (in particular, editing, modification and sampling of programs) involved laborious operations. In our view this consequence of separatism, characteristic of the developments of applied and systems programming, has at the current stage become a drag on the development of computer applications connected with the solution of complex problems.

5. This reasoning enables us to draw the conclusion that one of the principal directions of work in modern programming must be the creation of problem-oriented programming systems, ensuring adequate coverage of some applied activity. Without claiming to determine the package uniquely, and also taking into account the existence of other definitions, we would like to explain in rather more detail the terms used here [4].

We first note that every specific applied activity involves two factors. Firstly, the subject domain (that is, that to which the labour of the programmer is directed), representing the totality of applied problems solved by him, and secondly, the discipline of the work, that is, the totality of methods and rules accepted in processing, debugging and using programs. Adequacy of coverage means a sufficient completeness of facilities used in the course of a specific applied activity. In other words, both the range of models serving as material for the assembly of programs from a given subject field, and also the range of facilities for storing, editing modules, assembling programs and performing other types of work undertaken in the given applied activity must be sufficiently complete. In the general case the adequacy of coverage must be considered in time, that is, every variation in the applied activity must be taken into account by including in the package new applied or system programs. Therefore, we can speak of the "dynamic" adequacy of coverage.

If specific components of program provision are interpreted as indications of individual qualifications which the computer satisfies, then it is impossible not to acknowledge that at the present time the level of factual applied qualifications of the majority of computers is below the level of their potential applied qualifications, that is, it does not correspond to the rich store of applied programs of which we have already spoken. The fact is that unfortunately present applied programs are as a rule not supplemented by corresponding system possibilities, which hampers their use.

It follows from what was said above that the basic problem of the package problem at the present time must be regarded as the raising of the level of applied qualification of computers by the use of the methods of system programming based on the stores of applied programs already existing and being added to.

6. Organizationally a packet of applied programs may be represented as consisting of two parts: the functional and system contents.

The functional content reflects the specific subject field of the package and includes, for example:

the aggregate of modules used in the construction of programs for solving problems of a given subject field;

a collection of standard computing schemes whose defining modules consist of the solution program of some typical problem;

a collection of descriptions reflecting various functional and program-operational characteristics of the modules and standard computing schemes present in the package.

The requirements on the form of representation and organization of the elements of the functional content are usually established in the determination of the system facilities of the package.

The system content is the administrative organ of the package, reflecting the discipline of work with the package. The system content may for example include the following components:

the assignment language – the means of communication of the user with the package;

archives – the system of storage of the elements of the functional content and subsidiary information of the package;

the monitor – the totality of program facilities securing the operational possibilities of the package.

It should be noted that the system content can to a certain extent be invariant to the subject field, that is, the system content developed for some subject field may be easily adaptable for other subject fields.

7. The development of a package of applied programs involves the solution of a number of problems of which the following seem to us to be the most important.

The first problem, whose solution precedes the actual development of the package, consists of the definition of its problem orientation. Essentially, this problem reduces to the solution of two questions: the definition of the subject field, that is, the class of problems for which the given program package is intended, and the discipline of work with the package. The functional content of the package must be developed so as to ensure sufficiently complete material for the creation of programs from this subject field, and the system content must ensure the operations undertaken in its types. The totality of constructive elements and constructive facilities of the package also guarantees the raised qualification of the computer mentioned above.

Scarcely the least important problem encountered in the development of a package is the choice of its architecture, that is, the external form of the package, presented to the user. The architecture of a package is primarily reflected in the external specifications of its possibilities and in the description of the input language or languages. The successful solution of this problem ensures the efficient functioning and simple use of the package. Obviously, immediately after determining the architecture it is necessary to consider in detail and establish the fundamental principles of documentation of the package.

To exhibit the various types of moduli as constructive elements at all stages of functioning of the package, a modular analysis of the algorithms used in the given subject field, and also investigations of the general structure of the package must be carried out. Here we consider it necessary to emphasize that the concept of "module" can obviously be defined only as applied to a specific package. This concept in fact reflects the method of assembling programs assumed in the package, and therefore one can scarcely adhere to the traditional understanding of the module as some functionally completed program unit. The efficient realization of the principles of construction of programs from modules involves a variety of methods and facilities for the formal descritpion of the semantics and syntax of the modules. It must be mentioned that so far very few practically applicable solutions in this direction have been obtained.

The developers of a package must determine the collection of system facilities ensuring its efficient functioning. After this it is necessary to investigate the methods of realization of these facilities on the basis of the established operational security of the computer. It is essential that in the solution of these questions the simplicity of use and introduction of the package be taken into account, since practice shows that the degree of adoption of a system development is mainly decided by these factors.

A characteristic feature of modern advanced programming systems is the possibility of an interactive relationship with them, or as it is also usually expressed, the possibility of using them in a dialog mode. This mode of intercourse with the package is extremely promising, especially if it is considered that in some applied problems the dialog mode is the only practically applicable method of controlling a data-processing process. In this connection it is necessary to study the general principles of interaction of the user with the package of applied programs, to investigate the specific nature of the solution of problems in the dialog mode and determine its effect on the structure of the package. The results then obtained can be used for the development of specific languages of intercourse with the package and of system facilities for arranging the dialog.

Even the incomplete enumeration of the elements of the functional content of a package made above enables us to judge that it is practically impossible to attain a high degree of organization of work on such an extensive basis of data without corresponding means of information provision. In the creation of information provision it is necessary to investigate the means of describing the subject fields of the packages, the descriptions of the modules, including those for lexical purposes, and the descriptions of the computational environment in which the package is used. Considering that the use of information services always involves certain expenditures of time and storage, it is necessary to give special attention to the optimization of the information provision. Without detailing the possible approaches to the realization of an information service, we would nevertheless like to indicate that the use in it of only the means for a simple information search is obviously insufficient for the maintenance of such an extensive and dynamically variable basis of data as the functional content of a package.

We have already remarked that the fundamental problem of the package problem is improving the level of the applied qualification of the computer. Directly connected with this statement is the investigation and development of methods of automatic program construction based on a mathematical model of the problem to be solved, descriptions of the semantics of the module of the functional content and descriptions of the physical relations existing in the given subject field. In our view a completely automatic method of assembling programs should not be attempted. A more realistic and practical solution is a compromise in which the assignment of the package is formulated using both the facilities of the procedural description (that is, explicjtly defining all the data and the transformations to be carried out on them), and also the facilities of non-procedural description (that is, taking into account only the initial data and the final aim of some fragment of the process). With this approach the user can indicate to the computer exactly "what and how" it has to do. Moreover, he can present to it some independence in the choice of the method of solving individual subproblems, that is, assign to the package the automatic construction of individual parts of the program.

8. In ending the discussion of the individual aspects of the package problem, we would like to emphasize that the fundamental factor in the arrangement of operations on a package is the obligation to participate in the solution of each particular problem (in any case at the stage of formulating the problem and compiling the technical assignment to the program of the corresponding component of the package) of both the applied and systems programmers. Only with this approach is it possible to attain an organic interaction of the functional content with the system facilities and ensure high operational characteristics of the package.

In conclusion we would like to make the following observation. At the present time the package problem is living through the initial stage of its development. In a whole series of already existing program developments, which can be referred to this problem, the absence of a unified terminology, a difference in approaches and in understanding of the problems arising in the creation of the package of applied programs is observed. It therefore appears to be extremely urgent to require the definite isolation of the group of topics constituting the package problem, and the establishment of a unified terminology. We hope that this paper has enabled a definite step in this direction to be made.

Translated by J. Berry

REFERENCES

1. SAMARSKII, A.A. General questions of modular programming for problems of mathematical physics. In: *International Conference in "Structure and organization of packages of programs". Subjects of reports.* (Mezhdunar. konf. "Struktura i organizatsiya paketov programm". Tezisy dokl.) "Metsniereba", Tbilisi, 1976.
2. *The principal locations of the state stock of algorithms and programs* (Osnovnye polozheniya o gosudarstvennom fonde algoritmov i programm), VTs Akad. Nauk SSSR, Moscow, 1971.
3. The CPC Program Library. *Computer Phys. Communications,* **1**, 473–476, 1969.
4. GORBUNOV–POSADOV, M.M. KARPOV, V.Ya., KORYAGIN, D.A., KRASOTCHENKO, V.V. and SAMARSKII, A.A. An approach to the automation of the programming of a numerical experiment. In: *International conference "Structure and organization of packages of programs". Subjects of reports.* (Mezhdunar. konf. "Struktura i organizatsiya paketov programm". Tezisy dokl.) "Metsniereba", Tbilisi, 1976.