On an Integrated Computational Environment for Numerical Algebra

Valery Il'in(✉)

Institute of Computational Mathematics and Mathematical Geophysics SB RAS, Novosibirsk State University, Novosibirsk, Russia
ilin@sscc.ru

Abstract. This paper describes the conception, general architecture, data structure, and main components of an Integrated Computational Environment (ICE) for the high-performance solution of a wide class of numerical algebraic problems on heterogeneous supercomputers with distributed and hierarchical shared memory. The tasks considered include systems of linear algebraic equations (SLAEs), various eigenvalue problems, and transformations of algebraic objects with large sparse matrices. These tasks arise in various approximations of multidimensional initial boundary value problems on unstructured grids. A quite large variety of types of matrices, featuring diverse structural, spectral, and other properties are allowed; there can also be a wide diversity of algorithms for computational algebra. There are relevant issues associated with scalable parallelism through hybrid programming on heterogeneous multiprocessor systems, MPI-processes, multithread computing, and vectorization of operations, including those without formal constraints on the number of degrees of freedom and on the number of computing units. The numerical methods and technologies are implemented in the KRYLOV library, which provides the integrated subsystem of the ICE. There are various technical requirements imposed upon the software: extendibility of the set of problems and algorithms, adaptation to the evolution of supercomputer architecture, ability to reuse external products, and coordinated participation of development groups taking part in the project. The end goal of these requirements is to provide a product featuring a long life cycle, high performance, and general acceptance among end users of diverse professional backgrounds.

Keywords: Computational algebra · Sparse matrix · Iterative method · High performance · Krylov subspaces · Scalable parallelism · Heterogeneous supercomputer · Domain decomposition

1 Introduction

Solving systems of linear algebraic equations (SLAEs) is by far the main task in computational algebra, which is, in turn, the key stage when solving problems of mathematical modeling of processes and phenomena. After discretization, approximation, and linearization are carried out, all the diversity of initial

L. Sokolinsky and M. Zymbler (Eds.): PCT 2019, CCIS 1063, pp. 91–106, 2019.
https://doi.org/10.1007/978-3-030-28163-2_7

statements (e.g., classical and generalized, differential and/or integral, stationary and evolutionary, linear and nonlinear) leads to the necessity of solving SLAEs. This stage is a "bottleneck" in large-scale computational experiments where the number of resources required by applications grows, as a rule, nonlinearly with respect to the number of degrees of freedom and accounts for the main part in estimates of the computational complexity. Among other tasks of numerical algebra, we can mention eigenvalue problems, systems of nonlinear algebraic equations (SNLAEs), matrix equations, and various matrix and/or vector transformations.

Naturally, the main object of our attention is the solution of "large" and "difficult" systems of equations on modern heterogeneous multiprocessor computing systems (MPS) with distributed and hierarchical shared memory. A SLAE with a number of unknowns N of the order of 10^8 to 10^{11} is considered to be large, and a SLAE in which the matrices have condition numbers between 10^{12} and 10^{15} is considered to be difficult. We henceforth assume that arithmetic operations are performed with numbers in standard 64-bit double-precision floating-point format, which gives a relative error of about 10^{-15}. It is important to emphasize that with the advent of supercomputers capable of post-petaFLOPS performances and more, the task of solving SLAEs does not lose its relevance. Indeed, as a matter of fact, "appetite comes with eating" and, as supercomputer performances increase, both the order of current implementations of algebraic systems and their condition numbers also grow. One can confidently predict that, in the near future, double precision will not be sufficient for stable calculations. For example, the problem of using "smart" arithmetic operations with variable numbers of digits, including representations with numbers of bits both greater than 100 and less than 50, will become unavoidable.

SLAEs of considerable interest are those obtained after the original multidimensional boundary value problems for differential equations or their generalized variational statements are approximated by finite-difference methods, finite volume methods, finite element methods, or discontinuous Galerkin methods (FDM, FVM, FEM, or DGM, respectively) on unstructured grids. The matrices generated in this manner have two important features. First, they are sparse and banded, i.e. the total number of nonzero entries is relatively small ($\mathrm{NNZ} \ll N^2$) and, moreover, all of them are located in a band of width $M \ll N$ around the main diagonal. Secondly, their portraits have an irregular structure, i.e. the numbers of non-diagonal nonzero elements in the matrix lines can be specified only by enumeration. This leads to the need of saving the matrices in sparse compressed formats (for example, Compressed Sparse Row (CSR)) which rely on storing in memory only the nonzero entries and their numbers. This fact implies a significant specificity for the software implementations of the algorithms and slows down the process of access to the values of the matrix entries.

An important feature of SLAEs that significantly affects algorithms created for their solution is the block structure of the matrix. It is determined by the specific properties of the initial boundary value problem, the grid and approximation methods used, and by the ordering of the vector entries. The most complicated

matrices, in the structural sense, are those arising from grid approximations of multidimensional interdisciplinary problems described by systems of functional equations with many unknown vector functions (for example, density distributions of simulated substances present temperature, velocity, electromagnetic fields, etc.).

The main means for solving large systems are iterative methods, thanks to their less strict requirements on the amount of memory used and the number of computational operations needed. The most effective and common are the preconditioned algorithms in the Krylov subspaces. During decades, these algorithms have been used in active research. From the large amount of literature devoted to them, we will restrict our references in this paper to the monographs [1,2].

A special place in these issues corresponds to the problem of scalable parallelization of algorithms on MPS. Here, the main tool is the domain decomposition method (DDM), which has become one of the current divisions of computational algebra and to which special monographs, conferences, and Internet sites have been devoted (see [3,4]). The primary principle of the DDM is the partitioning of the original problem for a complex calculation domain into interconnected auxiliary problems for subdomains. Then, each subproblem can be synchronously solved approximately by a processor of a supercomputer. As for the parallelization technologies, the scaling is attained using MPI-processes, multithreaded computing, and operation vectorization. In a sense, multigrid approaches appeared as an alternative to the DDM (see [5] and the literature cited there). Those approaches give asymptotically optimal (from the theoretical standpoint) estimations of the computation volume which are inadequate for parallelization on MPS.

The study of iterative methods and their practical applications has recently been a matter of research in two major directions. The first of them is the development of efficient methods for preconditioning the initial SLAEs with the aim of improving their conditioning. To this end, numerous algorithms for approximate matrix vectorization have been deeply involved (see the reviews [1,2]). The second direction is the development of iterative processes assuming they are considered in Krylov subspaces. In this case, there are various approaches associated with a number of methods: deflation, aggregation, coarse-grid correction, low-rank matrix approximations, etc. [6,7].

Another important fact is that a quite large number of algebraic methods have already been implemented and are widely available over computer networks. A rather complete list of such methods is given in [8]. In this respect, it is worth mentioning the BLAS and SPARSE BLAS libraries of vector-matrix operations, containing, in particular, high-performance implementations on supercomputers of different types. The use of such standard functions in applications significantly increases, at large, the efficiency of the designed software.

The objective of this study is to develop the concept, architecture, data structure, and main components of an Integrated Computing Environment (ICE) for the high-performance solution of a wide class of SLAEs. We focus on the

creation of a product that can be effectively used in applied software systems and is capable of motivating a high demand among end users of diverse professional backgrounds. The considered mathematical software is a part of the KRYLOV library [9], which is a subsystem of the Basic System Modeling (BSM) [10], designed to support all major technological stages of mathematical modeling.

The paper is organized as follows. In Sect. 2, we classify the types of SLAEs considered in the paper. Sections 3 and 4 offer a brief presentation of methods for solving algebraic systems and existing software for their implementation. The next section describes the general structure and main components of the KRYLOV library. In the Conclusions section, we discuss plans for a further research on the development of an integrated tool environment for solving problems of computational algebra. For the sake of brevity, we often omit the references to terms and concepts from computational algebra that can be easily found on the Internet.

2 Classification of the Considered Tasks

From a formal point of view, we consider a trivial mathematical problem involving a SLAE:

$$Au = f, \quad A = \{a_{l,m}\} \in \mathcal{R}^{N,N}, \quad u = \{u_l\}, \quad f = \{f_l\} \in \mathcal{R}^N, \tag{1}$$

where, for simplicity, the matrix A of order N is assumed to be real, square, and positive definite in the sense of the condition

$$(Av, v) \geq \delta \|v\|^2, \quad \delta > 0, \quad (v, w) = \sum_{i=1}^{N} v_i w_i, \quad \|v\|^2 = (v, v). \tag{2}$$

However, symmetry is not required. Most algorithms considered below are applicable to more general SLAEs: complex (Hermitian or non-Hermitian), degenerate, or non-definite. A particular attention is given to sparse systems arising from the approximation of multidimensional boundary value problems by various methods, such as finite differences, finite volume, finite elements, and the discontinuous Galerkin algorithms on unstructured grids (see [8] and the literature cited therein). This type of grid equations is, for simplicity, written as

$$a_{i,i}u_i + \sum_{j \in \omega_i} a_{i,j}u_j = f_i, \quad i \in \Omega^h, \tag{3}$$

where ω_i denotes the set of off-diagonal nonzero entries placed on the i-th row of the matrix. On the other hand, sometimes matrices are presented in a block form,

$$A_{q,q}u_q + \sum_{r \in \Omega q}^{n} A_{q,r}u_r = f_q, \quad q = 1, \ldots, P, \tag{4}$$

$$A_{q,r} \in \mathcal{R}^{N_q, N_r}, \quad f_q \in \mathcal{R}^{Nq}, \quad N_1 + \ldots + N_P = N,$$

where Ω_q is the set of numbers of nonzero matrix computing blocks located in the q-th block row.

In the case of grid algebraic equations, the block representation of a SLAE can be visually associated with the geometric decomposition of the grid computation domain Ω to non-intersecting subdomains Ω_q:

$$\Omega = \Omega_1 + \ldots + \Omega_P, \quad \text{where} \quad \Omega_q \cap \Omega_r = 0 \quad \text{for} \quad q \neq r. \tag{5}$$

Relations (5) can also be interpreted as an algebraic decomposition if Ω_q is simply considered as a subspace of N_q induces corresponding to the u_q or f_q subvector.

The transformation of matrices to the block form is associated with renumbering vector components and matrix rows, which can generate a large number of computational methods and technologies. For example, based on the initial algebraic decomposition of a grid computational domain of the form of (5) without intersections, it is possible to form a decomposition with parametrized intersections for different numbers of common grid layers. Moreover, for large matrices, it is natural to consider their memory representation in a distributed way, both in the sense of various physical processors and at the logical level of MPI processes. Similar problems of renumbering and shaping various block structures also arise in efficient multigrid approaches to solving SLAEs (see the review in [5]).

The structural properties of the matrices under consideration strongly depend on the features of the initial boundary value problems and on the methods applied for their approximation. Typical block characteristics and matrix portraits can be classified according to the type of a system of equations for the corresponding applied problems: heat and mass transfer, hydro-gas dynamics, stress-strain state, multiphase filtration, electromagnetism, and others. It is also important to emphasize that, from theoretical and practical points of view, we are not interested in solving a specific SLAE on a fixed grid with a characteristic step h but a series of algebraic systems of the same type on a sequence of condensed (possibly nested) grids.

During last decades, approximations of higher orders have been made more exact, reaching an error in the numerical solution of the order of $O(h^\gamma)$, where $\gamma > 1$. As the order grows (up to $\gamma = 2, 4, 6, \ldots$), the number NNZ of nonzero matrix entries and the computational complexity of the algorithms increase, but the amount of memory required to ensure a given accuracy of the result significantly decreases. The last factor leads to a reduction in communications, and it is well known that communications not only slow down the overall computational process but also constitute the most energy-intensive operations.

From a technological point of view, it is also important to classify matrices according to the methods used f or storing them. It should be noted that, in addition to the universal CSR format, there are other common matrix representations and software converters from one format to another (see, for example, the Intel MKL library [11]). Let us mention, in particular, “small-block” formats, which are built similarly to CSR but store matrices of a fixed small size

instead of numeric entries of the matrix. It is also natural that more efficient formats can be created for some matrices of simple structure. For example, for a symmetric matrix, it suffices to store only the main diagonal and one of its triangular parts (above or below the main diagonal). In the case of a Toeplitz, a quasi-Toeplitz, or a band matrix, the problem of storing its nonzero entries is absolutely trivial. Finally, let us note a special type called the block-structured matrices, characterized by different storage methods for different blocks. Also, representations can be highly efficient when using quasi-structured grids [12], in which the computational domain is divided into subdomains with different types of grids, including regular and uniform ones.

We should also note such tasks as the solution of SLAEs at each time step using implicit approximations for the numerical integration of multidimensional initial boundary value problems. In [7], for example, it was shown that a special choice of initial approximations using the method of least squares significantly reduces the computational complexity.

3 Preconditioned Methods in Krylov Subspaces

In this section, we give a brief overview of modern iterative algorithms for solving SLAEs, their parallelization, and software implementations in public libraries.

3.1 Multi-conditional Methods for Semi-conjugate Directions

Preconditioned iterative methods in Krylov subspaces are currently the main approaches to solving large sparse SLAEs. By their computational complexity and resource-intensity, and by the methods of implementation, these methods are divided into two groups: for symmetric and for asymmetric algebraic systems. The first group includes the efficient conjugate gradient (CG) and the conjugate residual (CR) methods, based on short (two-term) recursions. To solve asymmetric systems, there are algorithms based on the biorthogonalization of the computed vectors, also using short recursions. Examples from this group are the BiCG and the BiCR methods, as well as their stabilized versions: BiCGStab and BiCRStab. These algorithms are less developed from the theoretical and practical standpoints. The most popular and reliable are the generalized minimal residual methods (GMRES or FGMRES), which have theoretically an optimal rate of convergence of iterations and high practical reliability (robustness), but are based on computations of long vector recursions.

We consider methods of semi-conjugate directions that are equivalent in the rate of convergence of iterations and are a direct generalization of the CG and CR algorithms for asymmetric SLAEs. We give a formal presentation of these iterative processes in a "multi-conditional" version with the possibility of using several preconditioning matrices at each step; the number of such matrices and their form may vary. These methods belong to the class of block methods since they use several directional vectors. Therefore, it is natural to consider the Krylov

subspaces generated by these vectors as blocks (see [13] and the references cited therein):

$$\begin{aligned} r^0 &= f - Au^0, \quad n = 0, \ldots, \quad u^{n+1} = u^n + P_n \bar{a}_n, \\ r^{n+1} &= r^n - AP_n\bar{a}_n = r^q - AP_q\bar{a}_q - \cdots - AP_n\bar{a}_n, \quad 0 \le q \le n, \\ P_n &= (p_q^n, \ldots, p_{M_n}^n) \in \mathcal{R}^{N,M_n}, \quad \bar{a}_n = (a_{n,1}, \ldots, a_{n,1})^\top \in \mathcal{R}^{M_n}. \end{aligned} \tag{6}$$

Here, $\bar{a}_n$ are the coefficient vectors and P_n is a matrix in which each of its M_n columns is a direction vector p_l^n associated with the corresponding precondition matrix $B_{n,l}$, $l = 1, \ldots, M_n$, whose form is not determined yet. Relation (6) has a remarkable property, namely if the directional vectors satisfy the orthogonality conditions

$$\begin{aligned} (Ap_k^n, A^\gamma p_{k'}^{n'}) &= \rho_{n,k}^{(\gamma)} \delta_{n,n'}^{k,k'}, \quad \rho_{n,k}^{(\gamma)} = (Ap_k^n,\ A^\gamma p_{k'}^{n'}), \\ \gamma &= 0, 1, \quad n' = 0, 1, \ldots, n-1, \quad k, k' = 1, 2, \ldots, M_n, \end{aligned} \tag{7}$$

then the following relations are valid for the residual functionals:

$$\Phi_n^{(\gamma)}(r^{n+1}) \equiv (r^{n+1}, A^{\gamma-1}r^{n+1}) = (r^q, A^{\gamma-1}r^q) - \frac{\sum\limits_{k=q}^{n}\sum\limits_{l=1}^{M_n}(r^q, A^\gamma P_l^k)^2}{\rho_{k,l}^{(\gamma)}}, \tag{8}$$

unless the coefficients are defined by the formulas

$$\alpha_{n,l} = \frac{\sigma_{n,l}}{\rho_{n,n}^{(\gamma)}}, \quad \sigma_{n,l} = (r^0, A^\gamma P_l^n), \tag{9}$$

in which case functional (8) attains a minimum in the block subspace

$$\mathcal{H}_n = \mathrm{Span}\{r^0, Ap_1^0, \ldots, Ap_{M_0}^0, \ldots, Ap_1^n, \ldots, Ap_{M_n}^n\}, \quad M = 1 + M_0 + \cdots + M_n, \tag{10}$$

if $\gamma = 1$. For a symmetric matrix A, the minimum of $\Phi_n^{(\gamma)}$ is attained for any value of γ.

The orthogonality conditions are satisfied, in particular, if the directional vectors are determined with the help of certain preconditioning non-degenerate matrices $B_{n,l}$ from the following recurrence relations:

$$\begin{aligned} p_l^0 &= B_{0,l}^{-1} r^0, \ \ p_l^{n+1} = B_{n+1,l}^{-1} r^{n+1} - \sum_{k=0}^{n}\sum_{l=1}^{M_k} \beta_{n,k,l}^{(\gamma)} p_l^k, \quad n = 0, 1, \ldots; \\ B_{n,l} &\in \mathcal{R}^{N,N}, \quad i = 1, \ldots, M_n; \ \gamma = 0, 1, \\ \bar{\beta}_{n,k}^{(\gamma)} &= \{\beta_{n,k,l}^{\gamma}\} = \left(\beta_{n,k,1}^{(\gamma)} \ldots \beta_{n,k,M_n}^{(\gamma)}\right)^T \gamma \in \mathcal{R}^{M_n}. \\ \beta_{n,k,l}^{(\gamma)} &= -\frac{\left(A^\gamma p_l^k, AB_{n+1,l}^{-1} r^{n+1}\right)}{\rho_{n,l}^{\gamma}}, \quad n = 0, 1, \ldots; \quad k = 0, \ldots, n; \quad l = 1, \ldots, M_n. \end{aligned} \tag{11}$$

In this case, formula (11) defines the multi-preconditioning of the Krylov subspace. In the case when the matrices A and $B_{n,l}$ in formulas (6)–(11) are symmetric, we obtain preconditioned methods for conjugate gradients and conjugate

residuals (MPCG and MPCR) for $\gamma = 0, 1$, respectively, whereas $\beta_{n,k,l}^{(\gamma)} = 0$ for $k < n$, and the direction vectors are computed from two-term recursions.

If $A \neq A^\top$, then, apparently, the most appropriate is the multi-preconditioned semi-conjugate residual method (MPSCR). In this case, when solving ill-conditioned asymmetric SLAEs associated with a large number of iterations due to the resource-intensity of long recursions (mainly due to an increase in the amount of memory used), they have to be shortened by force, which is done by either introducing the so-called restart procedure or limiting the number of orthogonalized directional vectors (or matrices with multi-preconditioning), or by applying both approaches simultaneously. In all these cases, the rate of convergence of the iterations drops, sometimes very noticeably, which is the inevitable price of saving memory.

To overcome this degrading effect, we will consider the application of the method of least squares (MLS) [6], restricting ourselves to using restarts in a "pure form". Assume, for simplicity, that the restarts are periodically repeated through the same number m of iterations. This means that at each iteration of number $n_t = mt$, $t = 0, 1, \ldots$, the residual vector is calculated not from the recurrence relations (6), but from the original equation, i.e.

$$r^{n_t} = f - Au^{n_t}. \tag{12}$$

Then, the recursion is used again in the usual way. More precisely, such an iterative process is conveniently described by the two-index notation of the corresponding numbers of consecutive approximate vectors

$$u^{n_t} = u^{t,0}, \quad u^n = u^{t,k}, \quad k = n - n_t \quad \text{for} \quad n \in [n_t, n_{t+1}].$$

Let us assume that we already know the values of the "restart" approximations $u^{n_0}, u^{n_1}, \ldots, n_0 = 0$. For the correction of the last iterative approximation, we will write a linear combination of the vectors

$$\begin{aligned} \hat{u}^{n_t} &= u^{n_t} + b_1 v_1 + \ldots + b_t v_t = u^{n_t} + V_t \bar{b}, \quad \bar{b} = (b_1, \ldots, b_t)^\top, \\ V_t &= \{v_k = u^{n_k} - u^{n_{k-1}},\ k = 1, \ldots, t\} \in \mathcal{R}^{N,t}, \end{aligned} \tag{13}$$

with coefficients b_n that will be found from the generalized solution of the overdetermined system obtained after multiplying Eq. (13) by the original matrix A:

$$W_t \bar{b} = r^{n_t} = f - Au^{n_t}, \quad W_t = AV_t. \tag{14}$$

There are several ways to solve SLAE (14): using SVD or QR decompositions for the matrix W_t, finding the generalized inverse matrix W_t^+ using the Greville formulas, calculating the normal solution by the method of least squares (using the left Gauss transformation, i.e. multiplying the common parts of Eq. (14) on the left by $W_t^\top$), or applying a "lightweight" (by condition number) transformation by reducing to the square system

$$V_t^\top A V_t \bar{b} = V_t^\top r^{n_t}. \tag{15}$$

In all these cases, after finding the vector $\bar{b}$ and carrying out the correction of the iterative approximation by formula (15), the next "restart" begins with the calculation of the residual

$$r^{n_t+1} = f - Au^{n_t+1}, \quad u^{n_t+1} = \hat{u}^{n_t}. \tag{16}$$

3.2 Parallel Domain Decomposition Methods

Historically, the DDM came into existence in the 19th century as an alternating Schwartz method for the theoretical study of geometrically complex boundary value problems by reducing them to simpler ones. If we turn from the differential equations to the grid ones, then, in algebraic terms, the initial decomposition of a domain into subdomains corresponds to the use of the Seidel block method. In due course, this method was transformed into the additive Jacobi–Schwartz block method, which is a natural way to parallelize the computational process on an MPS. Currently, both geometric and algebraic interpretations of the DDM exist on equal terms, mutually generalizing each of the approaches.

Referring to the block representation of SLAE (4), we can write down the iterative Jacobi–Schwartz method in the following form:

$$B_{q,q}u_q^{n+1} \equiv (A_{q,q}+\theta D_q)u_q^{n+1} = f_r + \theta D_q u_q^n - \sum_{r\in\Omega_q} A_{q,r}u_r^n, \quad r = 1,\ldots,P, \tag{17}$$

where $\theta \in [0,1]$ is the iterative parameter and D_q is the diagonal matrix defined by the relation

$$D_q e = \sum_{r\in\Omega_q} A_{q,r}e, \quad e = (1,\ldots,)^\top \in \mathcal{R}^{Nq}.$$

Let us turn to the geometric interpretation of this algorithm and emphasize that the solution of each q-th Eq. (12) is an independent solution of the auxiliary boundary value problem in the subdomain Ω_q. In this case, the formally introduced values θ and D_q correspond to the use of interface conditions between the contacting subdomains (see [13] for more details).

The iterative process (12) can be represented in the form:

$$u^{n+1} = u^n + B_1^{-1}(f - Au^n), \quad u^n = \{u_q^n\}, \quad B_1 = \text{block} - \text{diag}\{B_{q,q}\}, \tag{18}$$

where B_1 is the preconditioning matrix of this version of the DDM, which can be used in formulas (7), thereby generating preconditioned methods in the Krylov subspaces.

Let us now consider the possibility of accelerating the described block iterative Krylov-type methods based on the deflationary approach (see review in [14]). In this case, in addition to the conventional variational and/or orthogonal properties of computational successive approximations, supplementary conditions of orthogonality are imposed on them to the specially introduced m-dimensional fixed deflation subspace associated with a rectangular matrix

$$V = (v_1 \ldots v_m) \in \mathcal{R}^{N,m}.$$

For the sake of brevity, we will discuss the use of the deflation method as applied to the conjugate residual algorithm for solving a SLAE with a symmetric matrix A. First, there is an approach to optimizing, in a certain sense, the vector of the initial iterative approximation u^0. Let an arbitrary vector u^{-1} be given. Then, we define the vectors

$$u^0 = u^{-1} + Vc, \quad r^0 = r^{-1} - AVc. \tag{19}$$

The vector of unknown coefficients $c = (c_1, \ldots, c_m)^T$ in (19) is determined (assuming formally that $r^0 = 0$) from the solution of the overdetermined system

$$Wc = AVc = r^{-1}. \tag{20}$$

Applying the method of least squares (MLS) to (20), we find the normal solution

$$c = (W^\top W)^+ W^\top r^{-1},$$

providing a minimum norm of the residual vector:

$$\begin{aligned} r^0 &= T_0 r^{-1} \quad T_0 = I - W(W^T W)^{-1} W^T, \\ &(r^0,\ r^0) = (r^{-1},\ r^{-1}) - (W(W^T W)^{-1} W^T r^{-1},\ r^{-1}) \\ &= (W^T W z,\ z) - ((W^T W)^{-1} z,\ z),\ z = W^T r^{-1}, \end{aligned} \tag{21}$$

where the matrix T_0 is a symmetric (orthogonal) projector with the following properties:

$$T_0 = T_0^T = T_0^2, \quad W^T T_0 = T_0 W = 0,$$

that is, the space $\mathrm{Span}(W)$ belongs to the kernel $\mathcal{N}(T_0)$. Further, from the initial direction vector in the form

$$p^0 = r^0 - V(W^T W)^{-1} W^T A r^0 = B r^0, \quad B = I - V(W^T W)^{-1} W^T A, \tag{22}$$

we obtain the deflation orthogonality conditions

$$W^T r^0 = 0, \quad W^T A p^0 = 0, \tag{23}$$

for the vectors r^0 and p^0. Moreover, the introduced matrix B has the following easily verifiable orthogonal properties:

$$W^T AB = 0, \quad BV = 0. \tag{24}$$

Further iterations of the derived Deflated Conjugate Residual deflation algorithm are performed according to the "standard" formulas of the method of conjugate residuals with a preconditioning matrix, whose role in this case is played by B from (22):

$$\begin{aligned} &u^{n+1} = u^n + \alpha_n p^n, \quad \alpha_n = \sigma_n / \rho_n, \quad \rho_n = (Ap^n, Ap^n), \\ &r^{n+1} = r^n - \alpha_n A p^n, \quad \sigma_n = (ABr^n,\ r^n), \\ &p^{n+1} = B r^{n+1} + \beta_n p^n, \quad \beta_n = \sigma_{n+1} / \sigma_n. \end{aligned} \tag{25}$$

In this case, at each iteration, the minimum of the residual norm $\|r^n\|$ in the preconditioned Krylov subspace

$$\mathcal{K}_n(A,\ r^0, B) = \mathrm{Span}\big(r^0, ABr^0, \ldots, (AB)^{n-1}r^0\big),$$

and the computed vectors satisfy the orthogonality conditions

$$\begin{gathered}(ABr^k, r^n) = \sigma_n\delta_{k,n}, \quad (Ap^k, Ap^n) = \rho_n\delta_{k,n},\ k = 0, 1, \ldots, n-1, \\ W^T r^n = 0, \quad W^T Ap^n = 0,\ n = 0, 1, \ldots.\end{gathered} \tag{26}$$

Note that the introduced preconditioning matrix B is degenerate since $BW = 0$. Nevertheless, relations (24) and (26) ensure the orthogonality of all residual vectors to the kernel $\mathcal{N}(\bar{A}) = \mathrm{Span}(W)$ of the matrix $\bar{A} = AB$, thereby ensuring the convergence of iterative process (21)–(25).

The considered deflationary approach is quite universal and was studied in different ways under the names of aggregation methods, coarse-grid correction, and low-rank approximations of matrices (see [13]). Deflationary preconditioning matrices of the form B_z from (17)–(20), in particular, can be effectively used in conjunction with B_z in the multi-preconditioned semi-conjugate residuals (MP-SCR). In this case, we will in fact make use of the coarse-grid correction to accelerate the DDM, and each column of the matrix V from (14)–(17) corresponds to certain subdomains (see [13,14]).

On the whole, it can be said that there are a lot of versions of the DDM in Krylov subspaces, as well as a lot of adjoining multigrid methods, but here we do not dwell on them. For example, according to the methodology of quasi-structured grids in individual subdomains, auxiliary special-type SLAEs can be solved either by fast Fourier transform algorithms, or by optimal implicit alternating direction methods, or by modern adaptive incomplete factorization techniques, or by low-rank approximations of HSS matrices (hierarchical semi-separable approximations; see [15]), and so on.

4 On the Technology of Programming Parallel Algorithms

The final performance of computer-aided implementations of the SLAE solution methods obviously depends on two main factors: the mathematical efficiency of the application of the algorithms (which we already discussed in the previous section) and the adaptability of their mapping onto the supercomputer architecture, which determines the quality of the parallelization of a computing process of heterogeneous type with heterogeneous arithmetic devices (universal CPUs) or specialized accelerators of the type of GPGPU or Intel Phi, functioning with distributed or hierarchical shared memory.

In the framework of two-level methods for decomposing functioning domains in Krylov subspaces, scalable parallelization is generally achieved by means of hybrid programming. At the top level, the Jacobi–Schwartz block matrices for the subdomains with interprocessor exchanges are implemented in a distributed

way by organizing the MPI processes. At a lower level, the simultaneous solution of auxiliary SLAEs in the subdomains is performed on multi-core processors with shared memory, by using multithread computing. Additional acceleration can be accomplished using vectorization of operations in the AVX-type command system.

To ensure a high performance, several technological aspects need to be considered. First, to avoid idle processors that implement the solution of various auxiliary SLAEs, it is necessary to construct a balanced partition of the domain into subdomains, something that is not generally an easy task. Strictly speaking, all SLAEs in subdomains must be solved at the same time; this is highly problematic to attain on heterogeneous devices. For example, the equality of the dimensions of the algebraic systems does not ensure their being solved synchronously, even in case of identical calculators. The second point is associated with a decrease in communication losses. One of the possibilities of the task is associated with matching of data transfer and arithmetic operations in time, and the other, with the formation of information buffers and special management of communication operations. In general, it should be noted that the study and design of optimal computational schemes, including parallel ones, is mainly an experimental work whose success largely depends on the quality of planning and on the equipment used.

When developing a new generation of mathematical software, one cannot but bear in mind that there is already a huge number of publicly available libraries that implement computational algebra methods, including high quality ones and those adapted for modern supercomputer platforms, in which a considerable intellectual potential is incorporated.

The world scientific community engaged in numerical methods for linear algebra has historically turned out to be well organized, has its own regular journals and conferences, and has actively participated in contact groups throughout many years. What is even more important, these activities are not only committed with the development and theoretical analysis of new methods but also with the development of software and experimental research into the efficiency and performance of computer implementations on modern platforms.

The greatest success has been achieved in computational algebra problems involving relatively small matrices, both dense and sparse, corresponding to the BLAS and SPARSEBLAS libraries which have become standard and are widely used (for example, in the Intel MKL library, which features one of the most effective "direct solvers": PARDISO). Taking into account the current state of affairs, it can be said that the fundamental issues concerning software problems are basically close to each other, although research is steadily conducted to improve both algorithms and their implementations on newly emerging computer architectures.

Regarding the development and testing of methods for large problems with sparse matrices, the research in this case is carried out mostly in an academic style since it is associated with general high-tech problems of mathematical modeling, including the construction of grids, approximations of initial equations, and

others, that are related to indivisible stages of a single computing process. In particular, the construction of subdomains for large tasks must be started at the stage of generation of grids and distributed among processors, since the global matrix may not fit in the memory of a single computing node.

At the same time, libraries for solving large sparse SLAEs are available in fairly large numbers. Among the best known, we can mention SPARSE KIT, PETSc, HYPRE (see an informative review in [8]), and LParSol [16]. During the last decades, however, a noticeable trend in the development of software for mathematical modeling has been the creation of integrated computing environments focused on coordinating the participation of various groups in the design of products with a long life cycle, showing a steady development of functionality and adaptation to the evolution of computer architectures. To a certain extent, DUNE [17], INMOST [18], and BSM [10], which include libraries of algebraic solvers, can serve as examples of such developments.

5 Technical Requirements for the Creation of an ICE

An overview of the classes of tasks and methods for solving them suffices to see that the creation of mathematical software that claims to be an integrated computing environment for a broad range of problems is a big and complex process that requires significant investments and the involvement of highly qualified specialists from various fields. Obviously, such a knowledge-intensive and resource-demanding project should be planned on a strict methodological basis which can be defined through the following list of technical requirements for the content of research and development (R & D):

- A wide range of functionality with a regular support on modern scientific and technological levels. In our particular case, this implies a flexible expansion of the composition of solved SLAEs by means of different spectral, variational, structural, and other properties, and also with methods used and adapted to the specific features of the algebraic systems and information methods for their presentation and architectural specifications used by MPS. In particular, redundant functionality should generally provide for the selection of the best algorithm from the library to solve a specific problem to achieve highly contradictory features as of efficiency and universality.
- High performance of the code, with scalable parallelization of computations using hybrid programming tools, and without formal restrictions either on the number of degrees of freedom of the implemented tasks and algorithms or on the number of used computing nodes, cores, and other devices in heterogeneous supercomputers with distributed and hierarchical shared memory.
- Adaptability to the evolution of computer platforms and the multi-versatility of the functional core of the library, provided by modern component technologies and through the coordination of the external and internal working interfaces of the software-tool kernel.
- Universal and convertible data structures that are consistent with existing common formats and have the ability to reuse external software products that represent a high intellectual potential.

- Multilingual and cross-platform software functional content, ability to benefit from different styles of teamwork and openness for coordinated participation in the project of various groups of developers.
- Availability of user and internal interfaces that are both intelligent and friendly and focus on wide applications and active demand in diverse production areas by specialists of various professional backgrounds. In particular, it is necessary to distinguish between the support of effective activities of mathematicians-developers of new algorithms, information technologists (including experts in parallelization), and the end user, who needs only a minimum of information about the "backstage" activity of an ICE.

These architectural principles focus on the long life cycle of the product developed, as well as on the high productivity of the programming work of the participants in such a high-tech project. At the same time, the focus on supercomputing involves not only the speed of the algorithms but also the intelligence of the tool environment and skilled work with big data to avoid communication losses.

Naturally, the system infrastructure of the ICE must include the following library tools, necessary for the active development, maintenance, and efficient operation of a production software product:

- means for automated testing and comparative analysis of the efficiency of algorithms on characteristic SLAEs from methodological and practical problems, including the international matrix collections MATRIX MARKET, FLORIDA, BOEING, and others, as well as matrix generators for standard applications;
- user documents with descriptions of source data and features of the use of library algorithms, including examples (EXAMPLEs) of running solvers and recommendations on their choice for specific types of SLAEs, as well as archives of computational scenarios with the calculation results;
- configuration management tools and support for multi-version of the functional core of a library, providing a flexible expansion of the composition of computing modules and their adaptation to computer platforms;
- means of forming interfaces with developers, end users, and external software products, as well as the integration of the formed library modules with application software packages;
- regularly updated knowledge base on methods and technologies of computational algebra, containing up-to-date information with cognitive analysis and target search for scientific publications, software implementations, and sets of test examples.

Some prototypes of such developments, mainly of the information type, are the integration project "Tree of Mathematics" (www.mathtree.ru), supervised by the Siberian Branch of the Russian Academy of Sciences, and the ALGO WIKI project (www.parallel.ru), developed by Moscow State University.

It should be borne in mind that at the stages of development, verification, validation, and testing of computational library modules, as well as in the course of

trials or production operations, program codes are in different forms and require the corresponding methods of support. Different stages of product availability can be defined depending on the organization of the technological process, from a trial or an experimental version (alpha version) to a final product with high-quality performance indicators. An extensive range of features in the considered products, from the level of requirements of the modern scientific world to a high technological level, determines the hierarchy of qualifications of the project participants: from experts of academic background to specialists in supercomputing technologies.

It should be specially noted the importance of creating educational versions of an ICE, accompanied by cognitive techniques and relevant materials, aimed at students with university lecturers and professionals in advanced computer skills training courses.

6 Conclusions

The main result of this study is that we managed to give a concrete expression to conceptual propositions concerning the creation of mathematical software of new generation for the high-performance solution of a wide class of computational algebra problems. The software was constructed in the framework of an integrated computing environment focused on a long life cycle and steady development of a functional core and is able to adapt to the evolution of computer architectures. It has been designed to support the coordinated participation of various groups of developers and the active reuse of external software products. It intends to respond to the demand of end users from various professional areas. The system content of such a project is designed to significantly increase the productivity of programmers through intelligent automation tools for building parallel algorithms and their mapping onto a supercomputer architecture. In an industrial language, the considered ICE should ensure the transition to the production of the means of production in applied software systems as a basis for predictional mathematical modeling.

Acknowledgements. The work was supported by the Russian Foundation for Basic Research (grants No. 16-29-15122 and No. 18-01-00295).

References

1. Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd edn. SIAM (2003)
2. Il'in, V.P.: Finite Element Methods and Technologies. ICM&MG SBRAS, Novosibirsk (2007)
3. Dolean, V., Jolivet, P., Nataf, F.: An Introduction to Domain Decomposition Methods: Algorithms, Theory and Parallel Implementation. SIAM, Philadelphia (2015)
4. Domain Decomposition Methods. http://ddm.org
5. Shapira, Y.: Matrix-Based Multigrid: Theory and Application. Springer, Heidelberg (2008). https://doi.org/10.1007/978-0-387-49765-5

6. Il'in, V.P.: Two-level least squares methods in Krylov subspaces. J. Math. Sci. **232**, 892–902 (2018)
7. Il'in, V.: High-performance computation of initial boundary value problems. In: Sokolinsky, L., Zymbler, M. (eds.) PCT 2018. CCIS, vol. 910, pp. 186–199. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99673-8_14
8. Dongarra, J.: List of freely available software for linear algebra on the web (2006). http://netlib.org/utk/people/JackDongarra/la-sw.html
9. Butyugin, D.S., Gurieva, Y.L., Il'in, V.P., Perevozkin, D.V., Putuchov, A.V., Skopin, I.N.: Functionality and technologies of the algebraic solvers in the library Krylov. Herald SUSU Ser. Comput. Math. Inf. **2**(3), 92–103 (2013). (in Russian)
10. Il'in, V.P., Gladkih, V.S.: Basic system of modelling (BSM): the conception, architecture and methodology. In: Proceedings of the International Conference Modern Problems of Mathematical Modelling, Image Processing and Parallel Computing (MPMMIP & PC-2017), pp. 151–158. DSTU Publ. Rostov-Don. (2017). (in Russian)
11. Intel Math Kernel Library. Reference Manual. http://software.intel.com/sites/products/documentation/hpc/composerxe/enus/mklxe/mk-manual-win-mac/index.html
12. Il'in, V.P.: Mathematical Modeling. Part 1: Continuous and Discrete Models. SBRAS Publ., Novosibirsk (2017). (in Russian)
13. Il'in, V.P.: Multi-preconditioned domain decomposition methods in the Krylov subspaces. In: Dimov, I., Faragó, I., Vulkov, L. (eds.) NAA 2016. LNCS, vol. 10187, pp. 95–106. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57099-0_9
14. Gurieva, Y.L., Il'in, V.P.: On parallel computational technologies of augmented domain decomposition methods. In: Malyshkin, V. (ed.) PaCT 2015. LNCS, vol. 9251, pp. 35–46. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21909-7_4
15. Solovyev, S.: Multifrontal hierarchically solver for 3D discretized elliptic equations. In: Dimov, I., Faragó, I., Vulkov, L. (eds.) FDM 2014. LNCS, vol. 9045, pp. 371–378. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20239-6_41
16. Aleinicov, A.Y., Barabanov, R.A., Bartenev, Y.G.: An application of parallel solvers for SLAEs in the applied packages for engineering. In: Proceedings of the International Conference "Supercomputing and Mathematical Model", pp. 102–110. Unicef Publ. (2015). (in Russian)