Asynchronous Analogs of Iterative Methods and Their Efficiency for Parallel Computers

Maksym Savchenko, Alexander Chemeris

[1] *Institute for Modeling in Energy Engineering, 15 General Naumov str., Kyiv, Ukraine*

maxvell.ua@gmail.com, a.a.chemeris@gmail.com

Abstract. *This paper gives the comparison of some iterative methods (Jacobi method, Gauss–Seidel method and the Successive over-relaxation method) for solving systems of linear equations with their asynchronous implementations. Experiments show the effectiveness of asynchronous iterative methods for solving linear algebra problems on parallel multiprocessor systems.*

Keywords

Parallel processing, asynchronous iterative methods, linear algebra, effectiveness of calculation .

1 Introduction

Complex physical processes are commonly modelled by using methods based on solution of simultaneous linear algebraic equations (SLAE). It is indisputable that solution of SLAE is one of the major and prevailing problems in calculus mathematics. Iteration methods are extensively applied in using computing systems, although these methods are not effective when applying multiprocessor computers. Problems in implementing arise at the synchronization level of computation process, when some processors idle at each iteration, waiting for the other processors in the system to complete the computation for the current iteration and transmit the calculated values to the other processors. Iteration methods are based on initial guess, which are estimated before the beginning of calculating; during the calculating initial guess gradually approaches to the solution without accumulating errors, which is especially urgent for large SLAEs, when the total number of operations increases dramatically. After reaching the prescribed accuracy the calculating stops. As a rule, approximate equality of values at two adjacent iterations is a criterion for such a stop.

Asynchronous iteration methods were propositioned to minimize the idles of processors in computing system. In [1] the theoretical aspects of these methods were investigated, general conditions of asynchronous iteration process precision were given and estimation of convergence rate was obtained. However, for its practical implementation on a particular computational architecture with either shared or distributed memory it is necessary to investigate and specify the characteristics of the methods and, correspondingly, algorithms.

The present article deals with investigations carried out for evaluation of different asynchronous iteration methods , notably the fixed point iteration method (Jacobi), Gauss-Seidel method and successive overrelaxation method (SOR), in comparison with their synchronous analogs.

2 Organization of asynchronous iteration process

All above-mentioned methods were used for solution of the system of linear equations of the type

$$Ax = b\,, \tag{1}$$

where A – is square matrix of coefficients the left part of the simultaneous equations with $n \times n$, b – coefficient vector of the right part of the simultaneous equations, x – vector of unknowns.

Using computations, given in papers [2,3], formula for x_i^k calculation was obtained: for the Jacobi method – given in formula (2), for the Gauss-Seidel method – (3), for the SOR method – (4):

$$x_i^k = \frac{1}{a_{i,i}}\left(b_i - \sum_{j=1}^{n} a_{i,j} x_j^{k-1}\right), \tag{2}$$

$$x_i^k = \frac{1}{a_{i,i}}\left(b_i + a_{i,i} x_i^{k-1} - \sum_{j=1}^{n} a_{i,j} x_j^{k-1}\right), \tag{3}$$

$$x_i^k = (1-\omega) x_i^{k-1} + \frac{\omega}{a_{i,i}}\left(b_i + \sum_{j=1}^{i-1} a_{i,i} x_i^k - \sum_{j=i+1}^{n} a_{i,j} x_j^{k-1}\right), \tag{4}$$

where $i = 1, 2\ldots n$; ω – is the relaxation parameter.

Let's organize the following calculation process in multiprocessor computing system. Let each processor simultaneously compute its S component group of vector of unknowns. As this takes place, each processor uses data, realized at the moment in time, for generation of the result of its group. On completing the calculation of the next group approximation the processor transfers calculated values to computer memory and they become available for other processors.

Such approach is fair both for architectures with shared memory and for cluster computation systems. Altogether, asynchronous calculations can be presented as an expression [1]

$$x_i^T = q_i^T \varphi_i(x^{Ti}) + (1 - q_i^T) x_i^{Ti}, \; i = \overline{1,S}, \tag{5}$$

where $x_i^T \; (i = \overline{1,S})$ — are current approximation of unknowns vectors, realized at the moment in time T; $x_i^T = (0,...,x_i^T,0,...,0)'$, $x_i^{Ti} = (x_i^{T1}, x_i^{T2},...,x_i^{TS})'$ - are current approximation of unknowns vectors, found at the moment in time *Ti<T*, that mark the beginning of the next approximation of the component vector of unknowns group x_i^T; $\varphi_i(x^{Ti}) = \delta_i \varphi(x^{Ti})$, δ_i – diagonal matrix, diagonal components of which equate to figure of one if the correspondent component is calculates by the processor *i* and to figure of zero if otherwise; $q_i^T, i = \overline{1,S},$ – coefficients that equate to the figure of one if the time interval *T-Ti* is sufficient for the realization of the vector x_i^T and its storage to memory, and equale to zero otherwise.

Expression (5) is worthwhile only if at least one time interval *T-Ti* is sufficient for realization of the next approximation of the vector x_i^T, i.e. at least one coefficient q_i^T is equal to zero.

Correspondingly, asynchronous analogs of above mentioned methods organize in coordination with the following scheme. At the moment of completion of calculation of the next approximation of the component $x_i^{T-P_i^T}$ from the computer memory by any of the processors, this processor reads out the vector of the current approximation of unknowns, realized at the moment in time $T - P_i^T$. Then the equation $f_i(x_1^{T-P_i^T},...,x_{i-1}^{T-P_i^T}, x_i,...,x_{i+1}^{T-P_i^T},...,x_n^{T-P_i^T}) = 0$ is solved in regard to x_i, and after that we accept $x_i^T = x_i$, where T – is the time of termination of solution of the equation in regard to x_i. For the SOR method – $x_i^T = x_i^{T-P_i^T} + \omega_T (x_i - x_i^{T-P_i^T})$.

In consideration of peculiarities of the parallel computational environments as well as the character of the matrix of the coefficients of the unknowns *A*, using so called block method seems effective.

Let the given matrix *A* be divided into submatrices so that

$$A = \begin{bmatrix} A_{11} & \ldots & \ldots & \ldots & A_{1q} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ A_{q1} & \ldots & \ldots & \ldots & A_{qq} \end{bmatrix}, \tag{6}$$

and it is supposed that each matrix A_{ii} is invertible. In this case block Jacobi method of the solution of the system (1) that corresponds to the introduced decomposition, takes the form

$$A_{11}x_i^{k+1} = -\sum_{j \neq i} A_{i,j}x_j^k + b_i \ , \qquad (7)$$

where i = 1, …, q; k = 0, 1, …, the decomposition of vectors x и b conforms to the decomposition A. Hereby, implementation of one block Jacobi iteration requires the solution q of the systems of the type (7) with the matrixes of coefficients A_{ii}.

For example, asynchronous block Gauss-Seidel method is realized in the following way. The solution of the linear system (1) is distributed between R processors, each of which retrieves the unknowns of its equation subsystem. At the moment of completion of realization of the next approximation of the vector $(x^i)^{T-P_i^T}$ by the i processor, the components of this vector are written to the shared memory while the vector of the current approximation $x^{T-P_i^T}$, formed at the moment in time $T - P_i^T$, is read out of it. After that the i processor starts realizing new approximation of the vector x^i of the linear system

$$F_i((x^1)^{T-P_i^T},...,(x^i),(x^{i+1})^{T-P_i^T},...,(x^R)^{T-P_i^T}) \ . \qquad (8)$$

On completing the solution of this system we adopt the convention that $(x^i)^T = x^i$, where T — is the time of completing of the solution.

Inside the blocks the solution of systems can be implemented asynchronously, but in that case it is necessary to take into account a possibility of appearance of a situation when blocks of unknowns cannot get renewed before the moment of implementation of the next iteration. The latter depends on the level of balance of computational loading of processors (which requires that all A_{ii} matrices had the same size and structure), as well as on the way of data transmission.

3 The comparison of the calculation time of iteration methods and their asynchronous analogs

On the ground of formulas (2) – (4) were devised the algorithms of solution of the linear system in which iteration process must continue up to the moment when the values x_i^k become close to the values x_i^{k-1} with required accuracy ε.

For obtaining the experimental data we used matrix A with the size n=14 and absolute majority of nonzero elements, the vector X = {0, -1, 1, 1, …, 1} was chosen as initial guess x_i^0.

In the present article we investigated determinate analogs of asynchronous iteration methods, where at each iteration x_i^k were calculated in the following stages:

1) x_1^k, x_2^k in a conventional manner (based on x_i^{k-1});
2) x_3^{k+1}, x_4^{k+1} are calculated on the ground of $X^k = \{x_1^k, x_2^k, x_3^0, x_4^0\}$;
3) x_3^{k+2}, x_4^{k+2} are calculated on the ground of $X^{k+1} = \{x_1^k, x_2^k, x_3^{k+1}, x_4^{k+1}\}$;
4) x_3^{k+3}, x_4^{k+3} are calculated on the ground of $X^{k+2} = \{x_1^k, x_2^k, x_3^{k+2}, x_4^{k+2}\}$;
5) etc.

Each method and their asynchronous analogs were calculated five times for obtaining statistically accurate data. In the tables 1 and 2 are given the obtained results for different values of required accuracy ε, where *iter* is the number of iterations necessary for obtaining the required accuracy. On the fig. 1 is given the time T, necessary for obtaining the required accuracy for the investigated iteration methods. The values ε are given in logarithmic scale.

As we can see, using both synchronous and asynchronous implementation, the most efficient is Gauss-Seidel method. It will be observed that successive overrelaxation method (SOR) in the majority of cases proved to be less efficient. The latter can be explained by the difficulty in the choice of relaxation parameter ω. Using Jacobi method with high accuracy we didn't obtain results (the value of required accuracy ε wasn't obtained).

Tab. 1. The results of the experiments for the synchronous methods

$Log_{10}(\varepsilon)$	Jacobi method		Gauss-Seidel method		SOR method	
	iter	T, sec.	iter	T, sec.	iter	T, sec.
-8	-	-	16	0,905	16	1,035
-7	-	-	14	0,796	14	0,905
-6	-	-	12	0,671	12	0,835
-5	-	-	10	0,655	10	0,655
-4	26	1,779	9	0,530	9	0,648
-3	22	1,341	7	0,468	7	0,476
-2	16	0,936	6	0,328	6	0,437
-1	11	0,733	4	0,249	4	0,297

Tab. 2. The results of the experiments for the asynchronous methods

$Log_{10}(\varepsilon)$	Jacobi method		Gauss-Seidel method		SOR method	
	iter	T, sec.	iter	T, sec.	iter	T, sec.
-8	-	-	14	0,859	16	1,222
-7	-	-	12	0,834	14	1,115
-6	-	-	11	0,612	12	0,941
-5	-	-	9	0,541	10	0,536
-4	16	1,561	8	0,415	9	0,495
-3	12	1,209	6	0,392	7	0,518
-2	7	0,787	5	0,307	6	0,473
-1	4	0,597	4	0,206	4	0,381

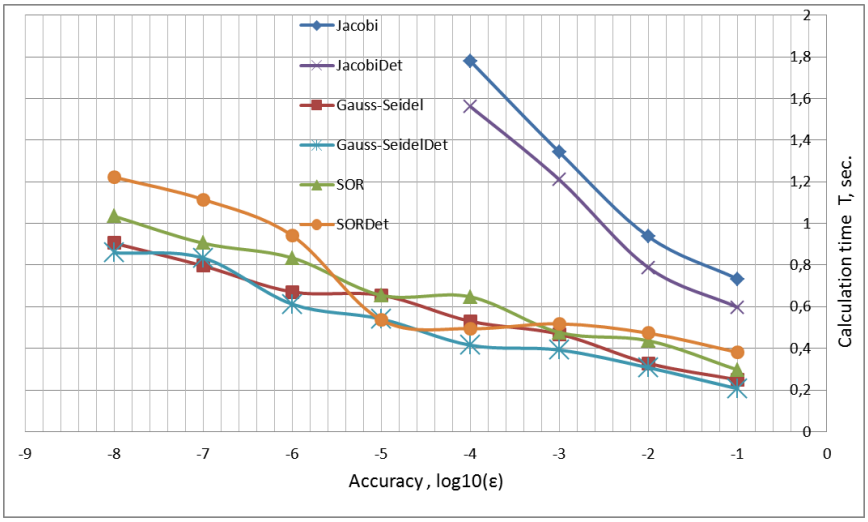

Fig. 1. The correspondence of the calculation implementation time and required accuracy ε for different iteration methods

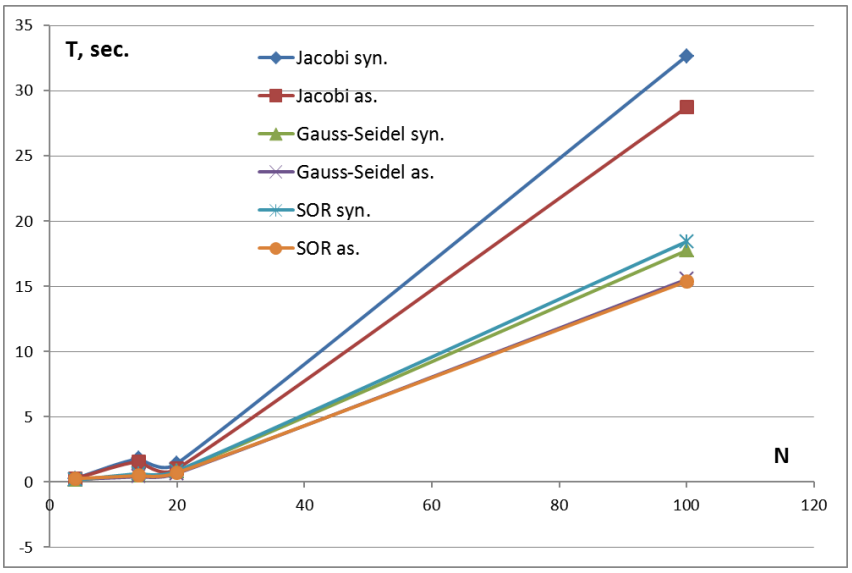

Fig. 2. The correspondence of the problem solution time and dimention of problem with the same ε.

4 Conclusion

Comparing each iteration method with its asynchronous analog it is obvious that in absolute majority of cases the latter provide computational speedup. As this takes place, when using asynchronous iteration methods, increasing the dimension of problem increases the computation time reduces. For each specific problem it is necessary to define which asynchronous analog of which method should be used, which computational scheme should be formed and how to organize the data transmission between processors of computational system.

The experiments that were carried out on quite simple examples show the efficiency of using the asynchronous iteration methods for the solution of linear algebraic problems on parallel multiprocessor computational systems. The present results were obtained in the process of the organization of a determinate computational process, when the number of iterations, computed on one processor between the exchange of data in a computational system, is assigned a priori. Organizing a completely asynchronous process can lead to a greater computational speedup.

References

[1]. V. N. Beletskyy «Multiprocessor and parallel structure to the organization of an asynchronous computation». In Russian– Kiev: Nauk. dumka, 1988. – 239 p.

[2]. S. A. Belov, N.Yu. Zolotykch “Laboratory workshop on numerical linear algebra”. In Russian – Nizhny Novgorod: Publishing house of the Nizhny Novgorod State University named. N.I. Lobachevsky, 2005. – 235p.

[3]. M.Yu. Savchenko, O.A.Chemerys "Asynchronous parallel implementation of the method of calculating the unsteady modes in the linear section of the pipeline", in Ukrainian - Collected Works of PIMEE of NASc of Ukraine. - Kyiv, 2011. - Vol. 61. - pp. 112-116.

[4]. *Gerard M. Baudet* Asynchronous Iterative Methods for Multiprocessors // Journal of the Association for Computing Machinery, Vol. 25, No. 2, April 1978, pp. 226-244.

[5]. *J. M. Ortega*, Introduction to Parallel and Vector Solution of Linear Systems, Plenum Press, 1988