Int. J. Appl. Math. Comput. Sci., 2009, Vol. 19, No. 2, 207–217
DOI: 10.2478/v10006-009-0017-3

amcs

INFLUENCE OF PRECONDITIONING AND BLOCKING ON ACCURACY IN SOLVING MARKOVIAN MODELS

BEATA BYLINA, JAROSŁAW BYLINA

Institute of Mathematics
Marie Curie-Skłodowska University, Pl. M. Curie-Skłodowskiej 1, 20–031 Lublin, Poland
e-mail: {beatas,jmbylina}@hektor.umcs.lublin.pl

The article considers the effectiveness of various methods used to solve systems of linear equations (which emerge while modeling computer networks and systems with Markov chains) and the practical influence of the methods applied on accuracy. The paper considers some hybrids of both direct and iterative methods. Two varieties of the Gauss elimination will be considered as an example of direct methods: the LU factorization method and the WZ factorization method. The Gauss-Seidel iterative method will be discussed. The paper also shows preconditioning (with the use of incomplete Gauss elimination) and dividing the matrix into blocks where blocks are solved applying direct methods. The motivation for such hybrids is a very high condition number (which is bad) for coefficient matrices occuring in Markov chains and, thus, slow convergence of traditional iterative methods. Also, the blocking, preconditioning and merging of both are analysed. The paper presents the impact of linked methods on both the time and accuracy of finding vector probability. The results of an experiment are given for two groups of matrices: those derived from some very abstract Markovian models, and those from a general 2D Markov chain.

Keywords: preconditioning, linear equations, blocking methods, Markov chains, WZ factorization.

1. Introduction

Computer networks are continuously extended. Users demand bands, they want to send in real time both picture and sound. That is why modeling network work becomes an important problem regarding both predicting its work and preventing breakdown. One of computer networks modeling methods are Markovian models. They need some information: in what states network can exist and what the probabilities (or rates) of transitions among states are. Based on the above, a system of equations is built to describe transition rates among particular states to determine the probability of a particular network's state occurrence. After solving the system of equations and obtaining state probabilities, it is possible to determine the parameters of network work (e.g., capacity, packets loosing probability, delay, packets dropping rate).

In the steady state (independent of time), while modeling with Markov chains, we obtain the following system of linear equations:

$$\mathbf{Q}^T\mathbf{x} = \mathbf{0}, \qquad \mathbf{x} \geq \mathbf{0}, \qquad \mathbf{x}^T\mathbf{e} = 1, \qquad (1)$$

where $\mathbf{Q}$ is the transition rate matrix, and $\mathbf{x}$ is a vector of state probabilities and the vector $\mathbf{e} = (1, 1, \ldots, 1)^T$. The matrix $\mathbf{Q}$ is an $n \times n$ matrix, usually a big one, of rank $n - 1$, sparse, with a dominant diagonal. It is also a singular matrix, thus demanding adequate methods to solve it. Solving Markovian models demands overcoming both numerical and algorithmic problems. Solving (1) generally demands applying iterative methods, projection methods or decomposition methods but, occasionally (for the need of an accurate solution), direct methods are used. Information concerning applying direct methods in Markov chains can be found in (Harrod and Plemmons, 1984; Golub and Meyer, 1986; Funderlic and Meyer, 1986; Funderlic and Plemmons, 1986). Iterative methods are described, for example, in (Jennings and Stewart, 1975; Stewart and Jennings, 1981), and projection methods are discussed in (Saad and Schultz, 1986; Bylina, 2003), while decomposition methods are described in (Schweitzer and Kindle, 1986; Haviv, 1987). A rich material concerning all methods mentioned above can be found in (Stewart, 1994).

Both direct and iterative methods have advantages and disadvantages. One of the ways to take advantage of both methods to solve big sparse problems is linking them together. This kind of approach is described in

(Duff, 2004), among others.

In our article—which is an improved and extended version of (Bylina and Bylina, 2007; 2008)—we will point at two ways of merging iterative and direct methods: preconditioning and blocking, as well as their linking. The novelty of this elaboration concerns applying the WZ factorization method instead of the LU factorization method—regarding both incomplete factorization in preconditioning techniques as well as during solving with the block Gauss-Seidel method. These methods are described in (Bylina and Bylina, 2007).

We would like to try to answer some questions about hybrid methods. Is there any point in linking iterative methods with direct methods? For what matrices is it worth doing? What are matrices' general properties that affect the behaviour of the algorithms? Is it advisable to apply preconditioning and blocking to an iterative method both at the same time? What is the influence of various methods and their combinations on accuracy?

The article is composed as follows: Section 2 includes a discussion on the features of direct and iterative methods used in solving linear equation systems derived from Markov chains. Section 3 presents the idea of preconditioning and incomplete factorizations as preconditioning techniques. Section 4 describes the block Gauss-Seidel method. Section 5 is devoted to the matrix group inverse, which is needed to find a condition number of singular matrices. Section 6 presents the results of a numerical experiment conducted for various matrices describing some Markov chains. In the last section there are conclusions regarding the conducted experiments.

2. Scope and constraints in the methods of solving systems of equations

In the literature we can find four approaches (mentioned above) concerning solving the system of linear equations (1). In the article we describe two methods: direct and iterative ones.

2.1. Direct methods. Direct methods (also known as accurate ones) are those which would lead to an accurate solution of the system of equations after a finite number of steps if all calculations were done without rounding. All direct methods are modifications of the Gauss elimination method, which enables determining LU factorization, where the matrix $\mathbf{L}$ is lower triangular and the matrix $\mathbf{U}$ is upper triangular.

The features of the direct methods are as follows:

- They give a solution in a finite number of steps by applying decomposition.
- In solving equations applying direct methods, elimination of one non-zero element in the reduction phase is often followed by creating some non-zero elements where earlier there were zeros. This fill-in effect not only makes matrix storage organization more complicated but also the size of the fill-in can be so big that the accessible memory ends fast.
- Direct methods enable determining the demanded number of operations before calculations (for the Gauss elimination: $n^3/3 + n^2/2 - 5n/6$ multiplications and $n^2/2 + n/2$ divisions).
- For some problems, solving with the use of direct methods provides a more precise answer more quickly than with the application of iterative methods.

WZ factorization is another version of the Gauss elimination (see Fig. 1), which was designed and described in (Evans and Hatzopoulos, 1979) especially for SIMD computers (*Single Instruction Stream—Multiple Data Stream*) in 1979. It was adapted to existing computer architectures (Evans and Barulli, 1998; Chawla and Khazal, 2003; Rao and Sarita, 2008). The paper (Yalamov and Evans, 1995) presents WZ factorization as the one that can be faster on computers with a parallel architecture than LU factorization for matrices with a dominant diagonal, which is the feature of transition rate matrices $\mathbf{Q}$.

Regarding the WZ factorization features mentioned above, this direct method was chosen for the system (1) to be solved. That method can be applied successfully to determining small numbers, where there is a need to implement calculations with a greater accuracy.

2.2. Iterative methods. Iterative methods start with some approximation and create a sequence of inaccurate results which converges—as expected—to a solution of the problem. It is difficult to know in advance how many iterations are demanded to gain assumed accuracy so it is impossible to determine the number of operations before finishing the calculations. The Gauss-Seidel method is an example of the iterative method and it is described below.

The matrix $\mathbf{Q}^T = \mathbf{L} + \mathbf{D} + \mathbf{U}$, where $\mathbf{L}$ is a strictly lower triangular matrix, $\mathbf{D}$ is a diagonal one and $\mathbf{U}$ is a strictly upper triangular one. Assuming that an approximation $\mathbf{x}^{(i)} = [x_1^{(i)}, \ldots, x_n^{(i)}]^T$ is given, in the Gauss-Seidel method the next approximation $\mathbf{x}^{(i+1)} = [x_1^{(i+1)}, \ldots, x_n^{(i+1)}]^T$ is determined to comply with the equation

$$a_{k1}x_1^{(i+1)} + \ldots + a_{kk}x_k^{(i+1)} + a_{k,k+1}x_{k+1}^{(i)} + \ldots + a_{kn}^{(i)} = b_k. \quad (2)$$

Transforming (2), we have the following result:

$$(\mathbf{L} + \mathbf{D})\mathbf{x}^{(i+1)} + \mathbf{U}\mathbf{x}^{(i)} = 0, \quad (3)$$

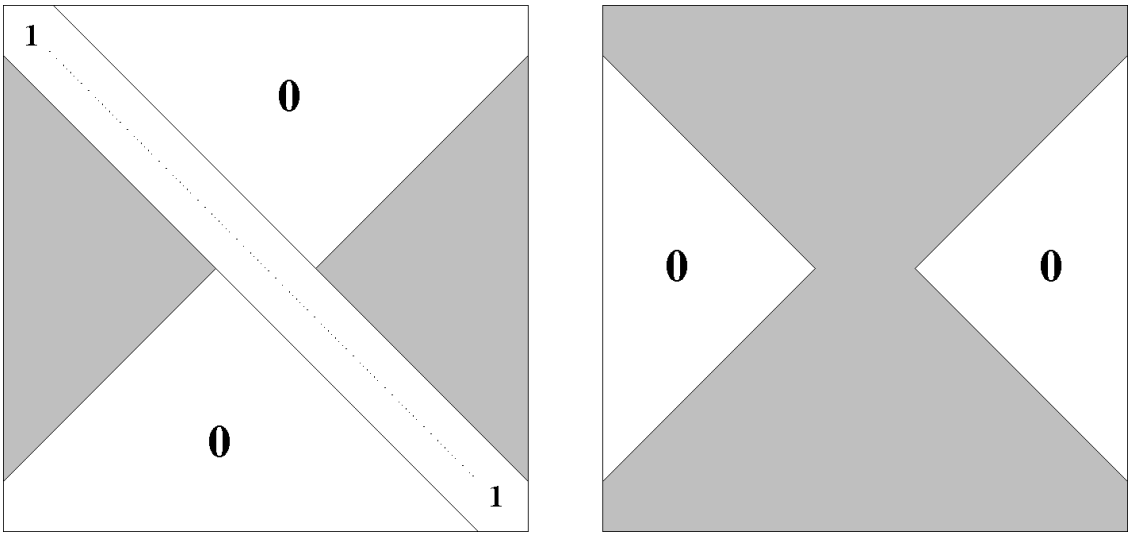

Fig. 1. Form of the output matrices in WZ factorization (left: $\mathbf{W}$; right: $\mathbf{Z}$).

which gives the following iterative formula, known as the Gauss-Seidel method,

$$\mathbf{x}^{(i+1)} = -(\mathbf{L}+\mathbf{D})^{-1}\mathbf{U}\mathbf{x}^{(i)}. \tag{4}$$

Equation (4) shows that in every iteration there are applied the newly computed vector values.

Advantages of iterative methods are as follows:

- they are easy to implement because there is no modification of matrices in the calculation process;
- they enable memory saving;
- the calculations can be done very precisely (the more precise they are, the longer the algorithm takes).

Problems concerning iterative methods include the following:

- difficulties with convergence for ill-conditioned matrices;
- sometimes a substantial number of iterations is required to arrive at convergence.

3. Preconditioning

The convergence rate of iterative methods depends upon the attributes of matrix $\mathbf{Q}$. The matrix $\mathbf{Q}$ is an ill-conditioned matrix, which can affect iterative methods applied to that matrix as they are slowly convergent or sometimes even divergent. One of the ways to prevent that is transforming the system of equations into an equivalent system, having the same solution but better attributes to solve it with iterative methods. So, preconditioning is an example of applying the above.

System (1) is transformed into the following:

$$\mathbf{M}^{-1}\mathbf{Q}^T\mathbf{x} = \mathbf{0}, \qquad \mathbf{x} \geq \mathbf{0}, \qquad \mathbf{x}^T\mathbf{e} = 1, \tag{5}$$

where the matrix $\mathbf{M}$ approximates the matrix $\mathbf{Q}^T$ in some way. So, instead of solving the system (1), we find a solution of the system (5). Applying any preconditioning is tightly connected with a high cost of calculations. It concerns the construction and application of the preconditioner (which means finding a matrix $\mathbf{M}$ approximating the matrix $\mathbf{Q}^T$ and inverting the matrix $\mathbf{M}$). The cost of preconditioner construction can be amortized by a lower number of iterations or by using the same preconditioner for different systems of equations.

Various techniques are applied to determine the matrix $\mathbf{M}$. It is very difficult to find good preconditioning to solve sparse systems of linear equations. It happens very often that theoretical results cannot be confirmed in practice. There are not enough tools and means to prove the above (besides numerical experiments).

Preconditioners are usually built based on the original matrix $\mathbf{Q}^T$. In (Benzi and Ucar, 2007), the authors consider preconditioned Krylov subspace methods for solving large singular linear systems arising from Markovian modeling. In the next section preconditioner construction will be described based upon incomplete factorizations applied to classical iterative methods.

3.1. Incomplete LU factorization. A preconditioning example based on incomplete LU factorization (marked with ILU) consists in the calculation of a lower triangular matrix $\mathbf{L}$ and an upper triangular matrix $\mathbf{U}$. The product of these matrices' product gives an approximating matrix $\mathbf{M}$. ILU has different variants but the simplest form of incomplete LU factorization is ILU(0), in which calculations are like in full LU factorization (that is, the Gauss

elimination) but only those elements l_{ij} and u_{ij} of the matrices $\mathbf{L}$ and $\mathbf{U}$ are stored for which adequate elements q_{ij} of the given matrix $\mathbf{Q}^T$ were non-zero. That is why the output matrices have the number of non-zero elements exactly the same as that for the matrix $\mathbf{Q}^T$. In that way the most important problem associated with sparse matrix factorization—the fill-in (non-zero elements in places where in the original matrix were only zeros, which is followed by the matrix change from sparse into a dense one and widening the room to store it)—is eliminated.

The matrix $\mathbf{Q}^T$ can be written as

$$\mathbf{Q}^T = \mathbf{LU} + \mathbf{R}_{LU}, \tag{6}$$

where the matrix $\mathbf{R}_{LU}$ is expected to be a small one (in a sense). Let

$$\mathbf{M} = \mathbf{LU}, \tag{7}$$

that is,

$$\mathbf{M}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}, \tag{8}$$

and then Eqn. (5) has the form

$$\mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{Q}^T\mathbf{x} = \mathbf{0}, \qquad \mathbf{x} \geq \mathbf{0}, \qquad \mathbf{x}^T\mathbf{e} = 1. \tag{9}$$

Let

$$\mathbf{S}_{LU} = \mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{Q}^T. \tag{10}$$

Then, Eqn. (9) has the following form:

$$\mathbf{S}_{LU}\mathbf{x} = \mathbf{0}, \qquad \mathbf{x} \geq \mathbf{0}, \qquad \mathbf{x}^T\mathbf{e} = 1. \tag{11}$$

Now, we can solve (11) with another algorithm, e.g., the Gauss-Seidel method.

3.2. Incomplete WZ factorization. Just like the incomplete LU factorization above, we can define IWZ (incomplete WZ factorization) more precisely: IWZ(0), whose details are given in (Bylina and Bylina, 2008).

In IWZ(0) there are remembered only those elements w_{ij} and z_{ij} of the matrices $\mathbf{W}$ and $\mathbf{Z}$ for which adequate elements q_{ij} of the given matrix $\mathbf{Q}^T$ were non-zeros. Hence, the output matrices have the number of non-zero elements accurately the same as that for the matrix $\mathbf{Q}^T$, and the fill-in is eliminated.

The matrix $\mathbf{Q}^T$ can be written as

$$\mathbf{Q}^T = \mathbf{WZ} + \mathbf{R}_{WZ}, \tag{12}$$

where the matrix $\mathbf{R}_{WZ}$ is expected to be a small one (in a sense). Let

$$\mathbf{M} = \mathbf{WZ}, \tag{13}$$

that is,

$$\mathbf{M}^{-1} = \mathbf{Z}^{-1}\mathbf{W}^{-1}, \tag{14}$$

and then Eqn. (5) has the form

$$\mathbf{Z}^{-1}\mathbf{W}^{-1}\mathbf{Q}^T\mathbf{x} = \mathbf{0}, \qquad \mathbf{x} \geq \mathbf{0}, \qquad \mathbf{x}^T\mathbf{e} = 1. \tag{15}$$

Let

$$\mathbf{S}_{WZ} = \mathbf{Z}^{-1}\mathbf{W}^{-1}\mathbf{Q}^T. \tag{16}$$

Then, Eqn. (15) has the following form:

$$\mathbf{S}_{WZ}\mathbf{x} = \mathbf{0}, \qquad \mathbf{x} \geq \mathbf{0}, \qquad \mathbf{x}^T\mathbf{e} = 1. \tag{17}$$

Just like in Section 3.1, we can now solve (17) with another algorithm, e.g., the Gauss-Seidel method.

4. Blocking: Block Gauss-Seidel method

It is possible to divide the transition rate matrix into blocks and develop iterative methods based on that division. Generally, iterative block methods demand more calculations per iteration, which is recompensed by a faster convergence rate.

The homogeneous system of equations (1) is divided into K^2 blocks in the following way:

$$\begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} & \cdots & \mathbf{Q}_{1K} \\ \mathbf{Q}_{21} & \mathbf{Q}_{22} & \cdots & \mathbf{Q}_{2K} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{Q}_{K1} & \mathbf{Q}_{K2} & \cdots & \mathbf{Q}_{KK} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_K \end{bmatrix} = \mathbf{0}. \tag{18}$$

We introduce block splitting

$$\mathbf{Q}^T = \mathbf{D}_K - (\mathbf{L}_K + \mathbf{U}_K), \tag{19}$$

where $\mathbf{D}_K$ is a block-diagonal matrix, $\mathbf{L}_K$ is a strictly block lower triangular one, $\mathbf{U}_K$ is a strictly block upper triangular one:

$$\mathbf{D}_K = \begin{bmatrix} \mathbf{D}_{11} & 0 & \ldots & 0 \\ 0 & \mathbf{D}_{22} & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & \mathbf{D}_{KK} \end{bmatrix}, \tag{20}$$

$$\mathbf{L}_K = \begin{bmatrix} 0 & 0 & \ldots & 0 \\ \mathbf{L}_{21} & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ \mathbf{L}_{K1} & \mathbf{L}_{K2} & \ldots & 0 \end{bmatrix}, \tag{21}$$

$$\mathbf{U}_K = \begin{bmatrix} 0 & \mathbf{U}_{12} & \ldots & \mathbf{U}_{1K} \\ 0 & 0 & \ldots & \mathbf{U}_{2K} \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 0 \end{bmatrix}. \tag{22}$$

The block Gauss-Seidel method is given by

$$(\mathbf{D}_K - \mathbf{L}_K)\mathbf{x}^{(i+1)} = \mathbf{U}_K\mathbf{x}^{(i)}. \tag{23}$$

Describing the equation mentioned above in a scalar-like form, we get

$$\begin{aligned} \mathbf{D}_{jj}\mathbf{x}_j^{(i+1)} = &\sum_{l=1}^{i-1} \mathbf{L}_{jl}\mathbf{x}_l^{(i+1)} \\ &+ \sum_{l=j+1}^{K} \mathbf{U}_{jl}\mathbf{x}_l^{(i)}, \end{aligned} \tag{24}$$

where $j = 1, 2, \ldots, K$ and $\mathbf{x}_j^{(i)}$ is the j-th sub-vector (of the size n/K) of the vector $\mathbf{x}^{(i)}$.

As a result, in each step we must solve K systems of equations, each of the size n/K, in the following form:

$$\mathbf{D}_{jj}\mathbf{x}_i^{(i+1)} = \mathbf{z}_j^{(i+1)}, \tag{25}$$

where

$$\mathbf{z}_j^{(i+1)} = \sum_{l=1}^{i-1} \mathbf{L}_{jl}\mathbf{x}_l^{(i+1)} + \sum_{l=j+1}^{K} \mathbf{U}_{jl}\mathbf{x}_l^{(i)} \tag{26}$$

and $j = 1, 2, \ldots, K$.

We can apply different direct and iterative methods to solve Eqn. (25). When using direct methods, for every block we make factorization only once before iterations, in the loop we change only right-hand sides of the equations. If the matrices $\mathbf{D}_{ii}$ have a special structure, for example, they are diagonal, upper triangular, lower triangular or tri-diagonal matrices, then LU or WZ factorizations are very simple to get and the block iterative method becomes very attractive. If the matrices $\mathbf{D}_{ii}$ do not have any of the mentioned structures, then it may appear that block solving demands iterative methods. Then, we have an inner iterative method within the main iterative method. It demands selecting a proper method as the inner one to be convergent and choosing an initial vector for the given method. There is a small number of outer iterations demanded to obtain convergence for a small number of blocks (the sub-matrices are big).

If we choose a direct method as the inner one, we have merging of an iterative method (Gauss-Seidel) and a direct method (WZ or LU), but their order is different.

5. Group inverse

The problem of solving a linear equation $\mathbf{Ax} = \mathbf{b}$ concerns the condition number $\kappa(\mathbf{A})$ usually defined as

$$\kappa(\mathbf{A}) = ||\mathbf{A}|| \cdot ||\mathbf{A}^{-1}||, \tag{27}$$

where $||\mathbf{A}||$ denotes any norm of the matrix $\mathbf{A}$.

If the solution of the equation $\mathbf{Ax} = \mathbf{b}$ is not very sensitive to small changes in $\mathbf{A}$ and $\mathbf{b}$, the matrix $\mathbf{A}$ (or the equation) is said to be well conditioned and it coincides with a low condition number. Otherwise, if the matrix has a high condition number, which means that the solution is sensitive to small changes in the coefficient matrix, the matrix (and the equation) is said to be ill conditioned. For ill conditioned matrices, the solutions obtained in a straight way from the equation $\mathbf{Ax} = \mathbf{b}$ could be very inaccurate.

In Markov chain problems, the infinitesimal generator matrix $\mathbf{Q}$ is singular, so its inverse does not exist. In such cases we cannot obtain a condition number from (27). However, then the concept of a group inverse or a generalized inverse plays a role analogous to that of the matrix inverse. The group inverse of $\mathbf{Q}$ is denoted by $\mathbf{Q}^{\#}$ and is defined as a unique matrix satisfying

$$\begin{aligned} \mathbf{QQ}^{\#}\mathbf{Q} &= \mathbf{Q}, \\ \mathbf{Q}^{\#}\mathbf{QQ}^{\#} &= \mathbf{Q}^{\#}, \\ \mathbf{QQ}^{\#} &= \mathbf{Q}^{\#}\mathbf{Q}. \end{aligned} \tag{28}$$

Thus, the condition number of a singular matrix $\mathbf{Q}$ can be defined as

$$\kappa(\mathbf{Q}) = ||\mathbf{Q}|| \cdot ||\mathbf{Q}^{\#}||. \tag{29}$$

It is shown in (Campbell and Meyer, 1979) that $\mathbf{Q}^{\#}$ fulfills the equation

$$\mathbf{I} = \mathbf{QQ}^{\#} + \mathbf{e}\pi, \tag{30}$$

where $\pi = \mathbf{x}^T$ is the stationary probability vector satisfying the equation $\mathbf{Q}^T\mathbf{x} = \mathbf{0}$.

For any nonsingular matrix, we can write $\mathbf{AA}^{-1} = \mathbf{I}$ as

$$\mathbf{A}[\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n] = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n], \tag{31}$$

and then

$$\begin{cases} \mathbf{Az}_1 = \mathbf{e}_1, \\ \mathbf{Az}_2 = \mathbf{e}_2, \\ \quad\vdots \\ \mathbf{Az}_n = \mathbf{e}_n, \end{cases} \tag{32}$$

where $\mathbf{e}_i = (e_{i_j})^T$, $e_{i_j} = \delta_{ij}$. For the nonsingular matrix $\mathbf{A}$, we can compute LU decomposition and then determine the columns of $\mathbf{A}^{-1}$:

$$\mathbf{z}_i = \mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{e}_i, \quad i = 1, 2, \ldots, n \tag{33}$$

and, analogously, we can write it with the use of WZ decomposition:

$$\mathbf{z}_i = \mathbf{Z}^{-1}\mathbf{W}^{-1}\mathbf{e}_i, \quad i = 1, 2, \ldots, n. \tag{34}$$

The algorithm presented below for the computation of the group inverse of an infinitesimal generator $\mathbf{Q}$ is based on this approach and on the fact that the group inverse of $\mathbf{Q}$ is the unique matrix that satisfies Eqns. (30) and (35) (which can be easily proved from (1) and (28) (Stewart, 1994)):

$$\pi\mathbf{Q}^{\#} = \mathbf{0}. \tag{35}$$

The algorithm for determining the group inverse with LU factorization was given in (Funderlic and Meyer, 1986). Here we present an analogous algorithm, but with the use of WZ factorization.

1. Compute WZ decomposition of $\mathbf{Q}$.
2. Compute the stationary probability vector π with the use of the zero determinant method (Bylina and Bylina, 2004).

3. For $i = 1, 2, \ldots, n$:

   - compute $\mathbf{z}_i = \mathbf{Z}^{-1}\mathbf{W}^{-1}(\mathbf{e}_i - \pi_i\mathbf{e})$; (Every $\mathbf{z}_i$ satisfies the equation $\mathbf{Q}\mathbf{Q}^{\#} = \mathbf{I} - \mathbf{e}\pi$, that is, $\mathbf{Q}[\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n] = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n] - \mathbf{e}[\pi_1, \pi_2, \ldots, \pi_n]$.)
   - compute $\mathbf{q}_i^{\#} = \mathbf{z}_i - (\pi\mathbf{z}_i)\mathbf{e}$.

4. Set $\mathbf{Q}^{\#} = [\mathbf{q}_1^{\#}, \mathbf{q}_2^{\#}, \ldots, \mathbf{q}_n^{\#}]$.

6. Numerical experiment

The implementation of the algorithms was done in the C language. Data structures to store the matrices $\mathbf{Q}^T$, $\mathbf{W}$, $\mathbf{Z}$, $\mathbf{L}$, $\mathbf{U}$ were full two-dimensional arrays situated in RAM. The numerical experiment was conducted on a PC computer with 1GB RAM and a Pentium IV 2.80 GHz processor. The algorithms were tested in the Linux environment, and the compilator `gcc` with the `-O3` option was used in the compilation.

The main goal of the tests was to check the behavior of preconditioning and blocking techniques and to compare them for some matrices and their influence on the accuracy of the results. The tests were carried out for some sparse matrices generated by the authors. The generated matrices described some Markov chains and they had all the attributes of the transition rate matrices $\mathbf{Q}^T$. Tables 1 and 4 show some information about the test matrices. In these tables, n means the number of columns and nz means the number of nonzero elements. It is worth noting that the condition numbers κ of all the presented matrices (in both tables) are very high—it is the reason why we should try to apply a technique enhancing the convergence of iterative methods.

6.1. Tests for abstract Markov chains. The matrices tested here differed in the average number of nonzero elements per column. Two matrices had more than 10 elements in a column (group I), and the other two had about four elements in a column (group II). That essentially affected calculations accuracy and iterative methods convergence.

The matrices were selected in the way allowing us to check the algorithms' behavior on various matrices, especially regarding their sparseness—it was caused by an observation of (Benzi and Ucar, 2007), where the convergence of some algorithms, such as GMRES (Saad and Schultz, 1986), for a matrix with the greatest density ($nz/n > 8$) was much worse.

The vector of probabilities π was determined for each tested matrix applying the usual Gauss-Seidel iterative method (GS) and the block Gauss-Seidel, the method where the blocks were solved using LU factorization (GSLU) and WZ factorization (GSWZ). For the block methods, K^2 was the number of blocks the transition matrices were divided into. Table 2 shows the time needed for the algorithms (GS, GSWZ, GSLU) as well as the impact of the above algorithms on result accuracy. The symbol i(e–$*$) means the number of iteration steps needed to get the accuracy e–$*$.

Applying block GS (in the variants GSLU and GSWZ) when to classic GS resulted the following observations:

- For all the matrices dividing into a small number of blocks gives faster method convergence. But dividing into a great number of blocks makes the accuracy to be obtained at the same number of steps as for the usual GS.
- The time of algorithm calculating depends only upon the number of blocks, independently of the groups of matrices. The smaller the number of blocks, the longer the time of calculations, which is normal for big matrices solved with accurate methods, where cost depends on size. Division into a large number of blocks shortens the time of calculations. The more blocks, the more similar the solving time to the usual Gauss-Seidel method.
- Calculation accuracy does not depend on factorization: WZ or LU. But factorization influences the time of the calculation. Dividing the matrices into a large number of blocks, we get a 1.5 time faster algorithm GSWZ than GSLU.
- The accuracy did not depend on the condition number, but rather on the matrix sparsity—sparser matrices gave worse results.

Incomplete factorizations IWZ and ILU were applied for the matrix $\mathbf{Q}^T$ as preconditioning. The traditional Gauss-Seidel method was applied for the obtained equation systems, and so was the block Gauss-Seidel method.

Table 3 shows the comparison of the methods: the traditional Gauss-Seidel algorithm (GS), preconditioned GS—both with IWZ (IWZ+GS) and with ILU (ILU+GS), and block GS, where the blocks were treated with WZ (GSWZ(K)) and with LU (GSLU(K)). Here, the number of blocks is K^2. $K = 2$ was chosen because of the best accuracy. The time was compared, as well as the convergence after 5, 10 and 20 iterations.

Some selected results from Table 3 are shown in Figs. 2 and 3.

In those tests we also linked together the iterative method with both preconditioning and blocking. The results are as follows:

- The convergence of matrices from group II (more sparse matrices) is weak so they need a larger number of iterations, or applying preconditioning or blocking.

Table 1. Test matrices attributes for some abstract Markov chains (Section 6.1).

group	ID	n	nz	average nonzeros per column (nz/n)	magnitude of condition number
I	1	1000	12678	12.68	e+09
	2	4000	42041	10.51	e+09
II	3	1500	5873	3.92	e+08
	4	4000	16946	4.24	e+09

Table 2. Comparison of algorithms calculation time (GS, GSWZ, GSLU) and the impact of the algorithms on result accuracy. The given time is for 20 iterations. The symbol i(e–$*$) denotes the number of iterations needed to achieve the given accuracy.

ID(n)		GSWZ		GSLU		GS
		$K=2$	$K=20$	$K=2$	$K=20$	
1(1000)	time:	6.59	0.29	11.57	0.3	0.18
	i(e–05):	4	6	4	6	6
	i(e–10):	8	12	8	12	13
	i(e–15):	12	18	12	18	19
2(4000)	time:	376.95	4.92	709.93	5.87	2.13
	i(e–05):	4	6	4	6	6
	i(e–10):	8	12	8	12	13
	i(e–15):	12	19	12	19	20
3(1500)	time:	21.68	0.58	40.13	0.63	0.33
	i(e–05):	7	11	7	11	11
	i(e–10):	17	>20	17	>20	>20
	i(e–15):	>20	>20	>20	>20	>20
4(4000)	time:	347.75	4.71	647.8	5.6	1.86
	i(e–05):	6	10	6	10	10
	i(e–10):	15	>20	15	>20	>20
	i(e–15):	>20	>20	>20	>20	>20

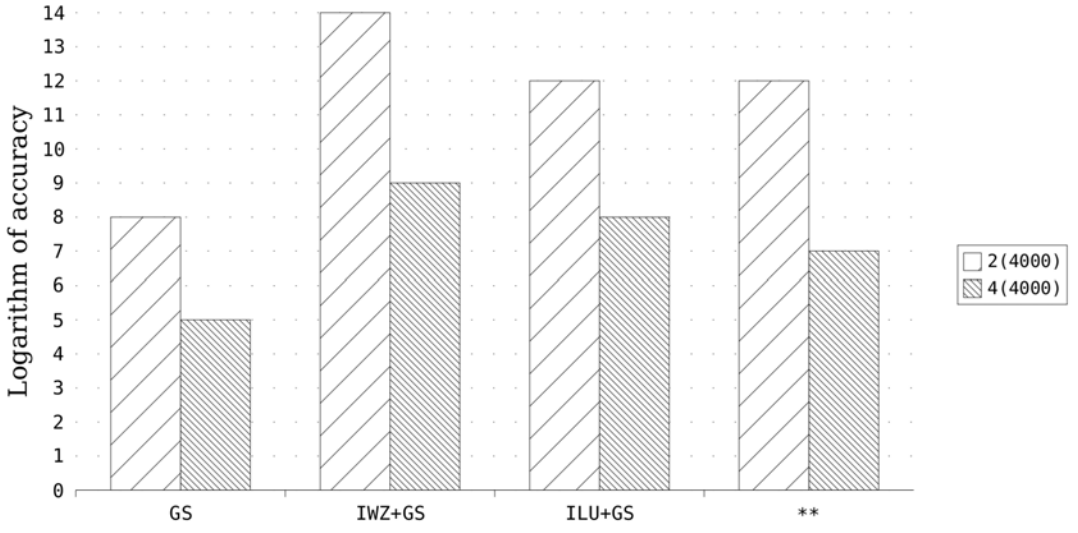

Fig. 2. Accuracy of the investigated algorithms for two selected matrices (2 and 4) from Section 6.1 and $i = 10$. The double asterisk ** denotes all the other algorithms (GSWZ(2), GSLU(2), IWZ+GSWZ(2), IWZ+GSLU(2), ILU+GSWZ(2), ILU+GSLU(2)), because they have the same accurcy for those matrices and $i = 10$.

- Applying either preconditioning or blocking improves the convergence of GS iterative methods.
- Preconditioning accelerates the convergence of iterative methods, especially using incomplete IWZ fac-

Table 3. Comparison of the time of algorithms (GS, IWZ+GS, ILU+GS, GSWZ, GSLU and their combinations) and their accuracy. The given time is for 20 iterations, the accuracies are given for i iterations (5, 10 and 20).

ID(n)	algorithm	time	$i = 5$	$i = 10$	$i = 20$
1(1000)	GS	0.18	e–04	e–08	e–16
	IWZ+GS	0.18	e–08	e–12	e–18
	ILU+GS	0.18	e–06	e–11	e–16
	GSWZ(2)	6.59	e–06	e–13	e–15
	GSLU(2)	11.57	e–06	e–13	e–15
	IWZ+GSWZ(2)	6.95	e–06	e–11	e–16
	IWZ+GSLU(2)	11.62	e–06	e–11	e–16
	ILU+GSWZ(2)	6.98	e–06	e–11	e–16
	ILU+GSLU(2)	11.52	e–06	e–11	e–16
2(4000)	GS	2.13	e–04	e–08	e–15
	IWZ+GS	2.54	e–09	e–14	e–19
	ILU+GS	2.33	e–07	e–12	e–17
	GSWZ(2)	376.95	e–06	e–12	e–15
	GSLU(2)	709.93	e–06	e–12	e–15
	IWZ+GSWZ(2)	379.61	e–06	e–12	e–15
	IWZ+GSLU(2)	678.62	e–06	e–12	e–15
	ILU+GSWZ(2)	391.34	e–06	e–12	e–15
	ILU+GSLU(2)	704.34	e–06	e–12	e–15
3(1500)	GS	0.33	e–03	e–04	e–08
	IWZ+GS	0.38	e–05	e–09	e–16
	ILU+GS	0.34	e–04	e–08	e–15
	GSWZ(2)	21.68	e–04	e–07	e–12
	GSLU(2)	40.13	e–04	e–07	e–12
	IWZ+GSWZ(2)	21.89	e–04	e–08	e–16
	IWZ+GSLU(2)	39.41	e–04	e–08	e–16
	ILU+GSWZ(2)	21.99	e–04	e–08	e–16
	ILU+GSLU(2)	39.87	e–04	e–08	e–16
4(4000)	GS	1.86	e–03	e–05	e–08
	IWZ+GS	2.38	e–05	e–09	e–17
	ILU+GS	2.15	e–04	e–08	e–16
	GSWZ(2)	347.75	e–04	e–07	e–13
	GSLU(2)	647.18	e–04	e–07	e–13
	IWZ+GSWZ(2)	359.92	e–05	e–07	e–15
	IWZ+GSLU(2)	649.09	e–05	e–07	e–15
	ILU+GSWZ(2)	391.34	e–05	e–07	e–15
	ILU+GSLU(2)	704.37	e–05	e–07	e–15

torization for the usual GS regarding both the time of calculations and accuracy.

- The properties (especially the sparsity) of the matrices influence accuracy.
- Applying both preconditioning and blocking to the Gauss-Seidel method does not improve the efficiency more than applying only preconditioning or only blocking.

6.2. Tests for a general 2D Markov model. The algorithms were also tested for matrices describing a general two-dimensional Markov chain model. The particular example has been taken from (Ridler-Rowe, 1967; Pollett and Stewart, 1994). The model is discussed there and it was used to compare diferent solution methods in (Stewart, 1994; Benzi and Ucar, 2007).

The state of the chain is described as a two-dimensional vector. In the first dimension, the state variable assumes all values from 0 through N_x. In the second dimension, the state variable takes on values from 0 through N_y.

This two-dimensional Markov chain model allows transitions from any nonboundary state to adjacent states in fixed directions (chosen from among north, south, east, west, north-east, north-west, south-east, south-west). A

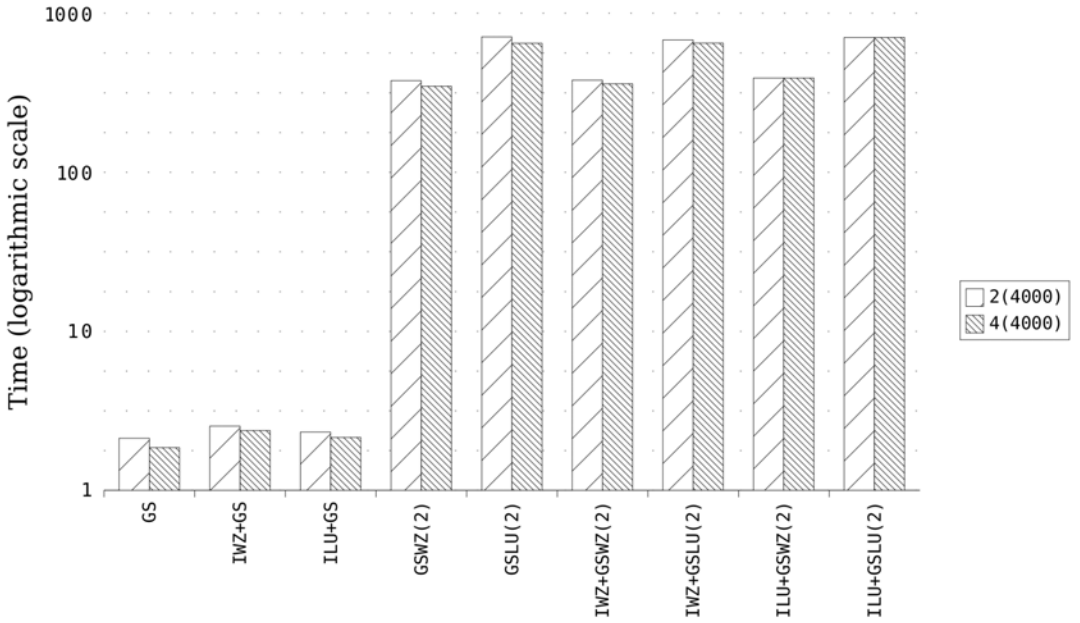

Fig. 3. Performance time (for $i = 20$) of the investigated algorithms for two selected matrices (2 and 4) from Section 6.1.

sample scheme of the model (with allowed directions: south, east and north-west) is shown in Fig. 4.

In the data set corresponding to this model, the values of N_x and N_y are shown in Table 4, along with the size of the matrix generated and the number of nonzeros in the matrix.

The accuracy of the results for this model is presented in Table 5. Here we tested only preconditioning, as it gives better results (see Section 6.1).

For these matrices, we can observe the following:

- The GS method has a very little accuracy for Matrix 5. However, preconditioning with ILU enhances the convergence a lot (IWZ is worse here).
- For Matrices 6 and 8 we have divergence for the traditional Gauss-Seidel method, but both the preconditioners yield convergence (ILU slightly better).
- The GS method for the Matrix 7 is convergent and both the preconditioners still enhance the convergence.

The matrices from Section 6.1 were denser (at least some of them) and had an undeniably better (i.e., smaller) condition number. On the other hand, the matrices presented in Section 6.2 were sparser and more ill conditioned, so they demanded some special treatment, such as preconditioning, because the Gauss-Seidel algorithm alone was not enough to find a solution quickly.

7. Conclusions

In the article the authors compared two techniques which can affect iterative calculations accuracy, i.e., the preconditioning and the blocking, in solving special linear equation systems. The coefficient matrices of those systems describe transition rates from one state to another of a model described by Markov processes.

The numerical experiment results show that for the investigated systems it is better to apply preconditioning than blocking to the Gauss-Seidel iterative method.

Moreover, applying both the enhancements to the original Gauss-Seidel method makes no difference, so it can be stated that preconditioning alone does suffice, giving a better performance time and accuracy.

For the first set of matrices (Section 6.1), both ways of linking direct methods with the Gauss-Seidel method give a similar improvement in the method convergence, but also, without it, the iterative algorithm alone would find a solution quite quickly (somewhat slower, though).

On the other hand, the second set (Section 6.2) seems to be more susceptible to the application of preconditioning (tests with blocking were not carried out for them because of the conclusions from the tests with the former set), and it corresponds to much higher condition numbers of the matrices. Without merging with direct methods, the Gauss-Seidel method alone has trouble finding stationary probabilities whatsoever.

All the matrices display one more interesting trait: regardless of the condition number, the relative density of a matrix (nz/n) influences the original convergence of the Gauss-Seidel method—the lower nz/n, the slower the

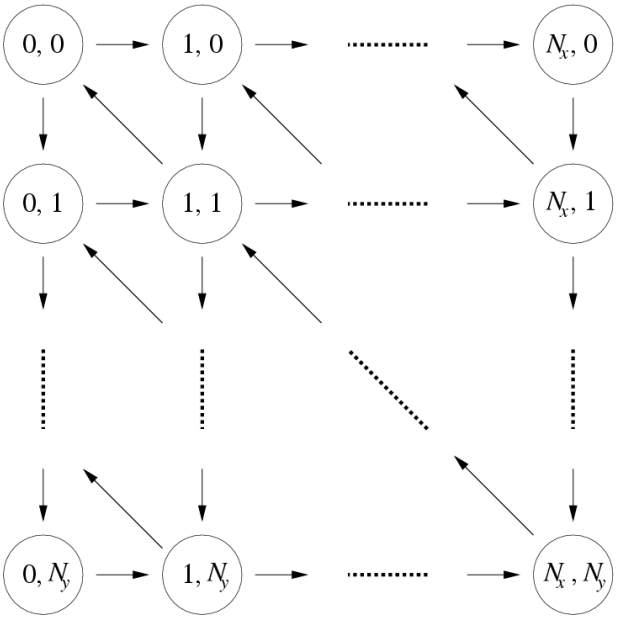

Fig. 4. Sample scheme of a two-dimensional Markov chain.

Table 4. Test matrices attributes for a 2D Markov model (Section 6.2).

ID	N_x	N_y	n	nz	average nonzeros per column (nz/n)	magnitude of condition number
5	64	16	1105	9457	8.56	e+12
6	45	45	2116	8281	3.91	e+15
7	45	45	2116	12376	5.85	e+18
8	64	64	4425	16641	3.76	e+18

Table 5. Comparison of the accuracy of the algorithms (GS, IWZ+GS, ILU+GS).

ID(n)	iteration number	GS	IWZ+GS	ILU+GS
5(1105)	1	e–02	e–02	e–02
	50	e–02	e–04	e–07
	100	e–03	e–06	e–14
6(2116)	1	e–02	e–02	e–02
	3	e–03	e–02	e–02
	12	e–02	e–02	e–02
	63	e–02	e–03	e–03
	100	e–02	e–04	e–07
7(2116)	1	e–02	e–02	e–00
	64	e–01	e–15	e–15
	100	e–08	e–15	e–15
8(4425)	1	e–03	e–03	e–02
	50	e–02	e–02	e–02
	100	e–02	e–03	e–04

convergence and, thus, the sparser the matrix, the more of a hybrid technique (namely, preconditioning) it needs.

And, finally, there are matrices for which applying incomplete WZ preconditioning produces better results, but there are also matrices for which it is worth using incomplete LU factorization.

References

Benzi, M. and Ucar, B. (2007). Block triangular preconditioners for M-matrices and Markov chains, *Electronic Transactions on Numerical Analysis* **26**(1): 209–227.

Bylina, B. and Bylina, J. (2004). Solving Markov chains with the WZ factorization for modelling networks, *Proceed-*

ings of the 3rd International Conference Aplimat 2004, Bratislava, Slovakia, pp. 307–312.

Bylina, B. and Bylina, J. (2007). Linking of direct and iterative methods in Markovian models solving, *Proceedings of the International Multiconference on Computer Science and Information Technology, Wisła, Poland*, Vol. 2, pp. 467–477.

Bylina, B. and Bylina, J. (2008). Incomplete WZ decomposition algorithm for solving Markov chains, *Journal of Applied Mathematics* **1**(2): 147–156.

Bylina, J. (2003). Distributed solving of Markov chains for computer network models, *Annales UMCS Informatica* **1**(1): 15–20.

Campbell, S. L. and Meyer, C. D. (1979). *Generalized Inverses of Linear Transformations*, Pitman Publishing Ltd., London.

Chawla, M. and Khazal, R. (2003). A new WZ factorization for parallel solution of tridiagonal systems, *International Journal of Computer Mathematics* **80**(1): 123–131.

Duff, I. S. (2004). Combining direct and iterative methods for the solution of large systems in different application areas, *Technical Report RAL-TR-2004-033*, Rutherford Appleton Laboratory, Chilton.

Evans, D. J. and Barulli, M. (1998). BSP linear solver for dense matrices, *Parallel Computing* **24**(5-6): 777–795.

Evans, D. J. and Hatzopoulos, M. (1979). The parallel solution of linear system, *International Journal of Computer Mathematics* **7**(3): 227–238.

Funderlic, R. E. and Meyer, C. D. (1986). Sensitivity of the stationary distrbution vector for an ergodic Markov chain, *Linear Algebra and Its Applications* **76**(1): 1–17.

Funderlic, R. E. and Plemmons, R. J. (1986). Updating LU factorizations for computing stationary distributions, *SIAM Journal on Algebraic and Discrete Methods* **7**(1): 30–42.

Golub, G. H. and Meyer, C. D. (1986). Using the QR factorization and group inversion to compute, differentiate and estimate the sensitivity of stationary distributions for Markov chains, *SIAM Journal on Algebraic and Discrete Methods* **7**(2): 273–281.

Harrod, W. J. and Plemmons, R. J. (1984). Comparisons of some direct methods for computing stationary distributions of Markov chains, *SIAM Journal on Scientific and Statistical Computing* **5**(2): 453–469.

Haviv, M. (1987). Aggregation/disagregation methods for computing the stationary distribution of a Markov chain, *SIAM Journal on Numerical Analysis* **24**(4): 952–966.

Jennings, A. and Stewart, W. J. (1975). Simultaneous iteration for partial eigensolution of real matrices, *Journal of the Institute of Mathematics and Its Applications* **15**(3): 351–361.

Pollett, P. K. and Stewart, D. E. (1994). An efficient procedure for computing quasi-stationary distributions of Markov chains with sparse transition structure, *Advances in Applied Probability* **26**(1): 68–79.

Rao, S. C. S. and Sarita (2008). Parallel solution of large symmetric tridiagonal linear systems, *Parallel Computing* **34**(3): 177–197.

Ridler-Rowe, C. J. (1967). On a stochastic model of an epidemic, *Advances in Applied Probability* **4**(1): 19–33.

Saad, Y. and Schultz, M. H. (1986). GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems, *SIAM Journal of Scientific and Statistical Computing* **7**(3): 856–869.

Schweitzer, P. J. and Kindle, K. W. (1986). An iterative aggregation-disaggregation algorithm for solving linear systems, *Applied Mathematics and Computation* **18**(4): 313–353.

Stewart, W. (1994). *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Chichester.

Stewart, W. J. and Jennings, A. (1981). A simultaneous iteration algorithm for real matrices, *ACM Transactions on Mathematical Software* **7**(2): 184–198.

Yalamov, P. and Evans, D. J. (1995). The WZ matrix factorization method, *Parallel Computing* **21**(7): 1111–1120.

Beata Bylina is a mathematics graduate (1998) and obtained her Ph.D. degree in computer science (*Algorithms to solve linear systems with the WZ factorization for Markovian models of computer networks*) in 2005 from the Institute of Theoretical and Applied Informatics of the Polish Academy of Sciences in Gliwice (Poland). She has been working at the Institute of Mathematics of Marie Curie-Skłodowska University in Lublin (Poland) since 1998, now as an assistant professor. Her research interests include numerical methods for systems of linear equations and for ordinary differential equations, scientific and parallel computing, and mathematical software.

Jarosław Bylina is a mathematics graduate (1998) and holds a Ph.D. degree in computer science (*Distributed methods to solve Markovian models of computer networks*), obtained from the Silesian University of Technology in Gliwice (Poland) in 2006. He has been working at the Institute of Mathematics of Marie Curie-Skłodowska University in Lublin (Poland) since 1998, now as an assistant professor. He is interested in numerical methods for Markov chains, the modelling of teleinformatic systems, as well as parallel and distributed computing and processing.

Received: 22 December 2007
Revised: 17 June 2008