



Linking of direct and iterative methods in Markovian models solving

Beata Bylina and Jarosław Bylina

Department of Computer Science
Institute of Mathematics
Marie Curie-Skłodowska University
Pl. M. Curie-Skłodowskiej 1, 20-031 Lublin, Poland
`beatas@hektor.umcs.lublin.pl`
`jmbylina@hektor.umcs.lublin.pl`

Abstract. An article identifies and assesses an effectiveness of two different methods applied to solve linear equations systems which result while modeling of computer networks and systems with Markov chains. The paper considers both the hybrid of direct methods as well as classic one of iterative methods. Two varieties of Gauss elimination will be considered as an example of direct methods: the LU factorization method and the WZ factorization method. Gauss-Seidel iterative method will be discussed. That issue points in preconditioning and matrix division into blocks where blocks will be solved applying direct methods. The paper presents an impact of linked methods on both time and accuracy of vector probability determining regarding particular networks and computer systems occurring.

1 Introduction

Computer networks are being continuously extended. Users demand bands, they want to send in real time both picture and sound. That is the reason for network work modeling to become important problem regarding both its work predicting and breakdown preventing. One of the computer networks modeling methods are Markovian models. Markovian models need two information; in what states can network exist and what are probabilities (or rates of transitions among states). Basing on mentioned above the equations system is being built to describe transition rates among particular states to determine the probability of particular networks states occurrence. After solving of equations systems linking states probabilities it is possible to determine the parameters of network work (ex.: capacity, packets losing probability, delay, packets dropping rate).

In steady state (independent of time) while modeling with Markov chains—we obtain by linear equation system like follows;

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

where \mathbf{Q} means transition rate matrix and \mathbf{x} is a vector of states probabilities and vector $\mathbf{e} = (1, 1, \dots, 1)^T$. Matrix \mathbf{Q} is the square matrix $n \times n$, usually a big one, of

rank $n - 1$, sparse, with dominant diagonal. It is also peculiar matrix demanding adequate methods to solve it. Markovian models solving demands overcoming both numerical and algorithmic problems. Equation system (1) solving generally demands applying of iterative methods, projection methods or decomposition methods but occasionally (for the need of accurate solution) direct methods are used. Information concerning direct methods applying in Markov chains can be found in the following elaborations: [7–10]. Iterative methods are described for example in [12, 13] and projection method are concerned in articles [2, 14]; and also decomposition methods were described in [11, 15]. The rich material concerning above mentioned methods can be found in [16].

Both direct methods and iterative ones have advantages and disadvantages. So one of the way to take advantage of both methods is linking them to solve big sparse problems. This kind of approach is described in Duff's paper [4] (among others).

In that article we will point at two ways of iterative and direct methods linking: preconditioning and block Gauss-Seidel method. Novelty of that elaboration concerns applying the WZ factorization method instead of LU factorization method —regarding both incomplete factorization in preconditioning techniques as well as during solving of block Gauss-Seidel method.

Second section of article includes discussion of direct and iterative methods features applied in solving of linear equation system modeling computer network work using Markov chains. Third section presents preconditioning idea and incomplete factorizations applying as preconditioning techniques. Fourth section describes block Gauss-Seidel method. Fifth section presents the results of numerical experiment conducted for matrices describing any computer network abstractive model. In last section there are conclusions regarding conducted experiments.

2 The scope and the constrains in the methods of equations system solving

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

2.1 Direct methods

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

The features of direct methods:

- they give solution in finite steps number by applying the decomposition;

- applying direct methods in equations solving – elimination of one matrix’s non-zero element in reduction phase is followed often by creating some non-zero elements where earlier there were zeros. The fill-in effect makes not only matrix organization saving more complicated but also the size of fill-in can be so big that accessible memory ends fast;
- direct methods enable determining of demanded operations number before calculations finishing (for Gauss elimination: $n^3/3 + n^2/2 - 5n/6$ multiplications and $n^2/2 + n/2$ divisions);
- for some problems solving with the direct methods determine more precise answer in shorter time than iterative methods.

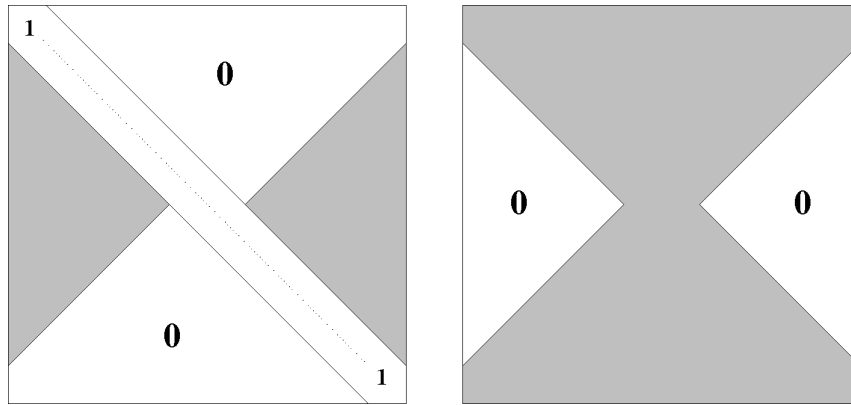


Fig. 1. The form of the output matrices in the WZ factorization (left: **W**; right: **Z**)

WZ factorization is another version of Gauss elimination (see Figure 1) which was designed and described in article [6] especially for SIMD computers (*Single Instruction Stream—Multiple Data Stream*) in 1979. For the reason that effective working computers were not constructed (of that architecture), factorization WZ was adapted to existing computer architectures [3, 5]. Paper [17] presents WZ factorization which can be faster on computers with parallel architecture from LU factorization for matrices with dominant diagonal—what is the feature of transition rate matrix **Q**.

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

2.2 Iterative methods

Iterative methods begin from some approximation and create indirect results sequence which tends—as it is expected—to solve the problem. It is difficult to

know in advance how many iterations is demanded to gain assumed accuracy so it is impossible to determine the number of operations before calculations finishing. Gauss-Seidel method is an example of iterative method and it is described further.

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

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

Transforming the formula we have got the following result:

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

what gives the following iterative formula known as Gauss-Seidel method:

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

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

Advantages of iterative methods:

- easy in implementation because there is no matrices modification in calculation process;
- they enable memory saving;
- calculation can be done very precisely (the more precisely they are done the longer time of algorithm making is).

Problems concerning iterative methods:

- problem with convergence for ill-conditioned matrices;
- sometimes great number of operations to get convergence.

3 Preconditioning

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

System (1) is transformed into the following formula system:

$$\mathbf{M}^{-1}\mathbf{Q}^T\mathbf{x} = \mathbf{0}, \quad \mathbf{x} \geq \mathbf{0}, \quad \mathbf{x}^T\mathbf{e} = 1 \quad (5)$$

where matrix \mathbf{M} approximates matrix \mathbf{Q}^T in some way. So, instead of system (1) solving, we find solution of system (5). Applying any preconditioning is tightly connected with high calculations cost. That cost concerns the construction and application of preconditioning. (what means finding matrix \mathbf{M} approximating matrix \mathbf{Q}^T and inverse matrix \mathbf{M}^{-1}). The cost of preconditioning construction can be amortized by iterations quantity or by using the same preconditioning for different equation systems. Different techniques are applied to determine matrix \mathbf{M} . It is very difficult to find good preconditioning to solve sparse linear equation system. It happens very often that theoretical results can not be confirmed by practice. There are not enough tools and means to prove above mentioned (besides numerical experiments). Preconditioners are usually built basing on original matrix \mathbf{Q}^T coefficients. In article [1] the authors consider preconditioning Krylov subspace methods for solving large singular linear systems arising from Markovian modeling. In next section preconditioning construction will be described basing upon incomplete factorizations applied for classical iterative methods.

3.1 Incomplete LU factorization

Preconditioning example basing on incomplete LU factorization—marked with ILU (incomplete LU) consisting in lower triangular matrix \mathbf{L} calculation and also upper triangular matrix \mathbf{U} which product gives 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 (taht is, Gauss elimination) but there are remembered only those elements l_{ij} and u_{ij} of matrices \mathbf{L} and \mathbf{U} for which adequate q_{ij} element of the given matrix \mathbf{Q}^T was non-zero. That is why output matrices have the number of non-zero elements accurately the same what \mathbf{Q}^T matrix has. In that way the most important problem associating sparse matrix factorization—fill-in (non-zero elements in places where in original matrix were only zeros—what is followed by matrices exchange from sparse for the dense one and widening the room to save it)—is eliminated.

Matrix \mathbf{Q}^T can be written as $\mathbf{Q}^T = \mathbf{L}\mathbf{U} + \mathbf{R}_{LU}$ where \mathbf{R}_{LU} matrix is expected to be a small one (in a sense). Let $\mathbf{M} = \mathbf{L}\mathbf{U}$, that is $\mathbf{M}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}$ and then equation (5) looks like

$$\mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{Q}^T\mathbf{x} = \mathbf{0}, \quad \mathbf{x} \geq \mathbf{0}, \quad \mathbf{x}^T\mathbf{e} = 1. \tag{6}$$

Let $\mathbf{S}_{LU} = \mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{Q}^T$. Then, equation (6) has the following form:

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

3.2 WZ incomplete factorization

Just like incomplete LU factorization above, we can define IWZ (incomplete WZ factorization)—more precisely: IWZ(0).

In IWZ(0) there are remembered only those elements w_{ij} and z_{ij} of matrices \mathbf{W} and \mathbf{Z} for which adequate q_{ij} element of the given matrix \mathbf{Q}^T was non-zero.

Hence, the output matrices have the number of non-zero elements accurately the same what \mathbf{Q}^T matrix has and the fill-in – is eliminated.

Matrix \mathbf{Q}^T can be written as $\mathbf{Q}^T = \mathbf{WZ} + \mathbf{R}_{WZ}$ where \mathbf{R}_{WZ} matrix is expected to be a small one (in a sense). Let $\mathbf{M} = \mathbf{WZ}$, that is $\mathbf{M}^{-1} = \mathbf{Z}^{-1}\mathbf{W}^{-1}$ and then equation (5) looks like

$$\mathbf{Z}^{-1}\mathbf{W}^{-1}\mathbf{Q}^T\mathbf{x} = \mathbf{0}, \quad \mathbf{x} \geq \mathbf{0}, \quad \mathbf{x}^T\mathbf{e} = 1. \tag{7}$$

Let $\mathbf{S}_{WZ} = \mathbf{Z}^{-1}\mathbf{W}^{-1}\mathbf{Q}^T$. Then, equation (7) has the following form:

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

4 Block Gauss-Seidel method

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

Homogeneous equations system is divided into K^2 blocks in the following way:

$$\mathbf{Q}^T\mathbf{x} = \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} & \dots & \mathbf{Q}_{1K} \\ \mathbf{Q}_{21} & \mathbf{Q}_{22} & \dots & \mathbf{Q}_{2K} \\ \dots & \dots & \dots & \dots \\ \mathbf{Q}_{K1} & \mathbf{Q}_{K2} & \dots & \mathbf{Q}_{KK} \end{bmatrix} [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_K]^T = \mathbf{0}$$

We introduce block splitting:

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

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

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

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

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

Block Gauss-Seidel method is given by:

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

Describing above mentioned equation in scalar-like form we get:

$$\mathbf{D}_{jj}\mathbf{x}_j^{(i+1)} = \left(\sum_{l=1}^{i-1} \mathbf{L}_{jl}\mathbf{x}_l^{(i+1)} + \sum_{l=j+1}^K \mathbf{U}_{jl}\mathbf{x}_l^{(i)} \right) \quad \text{for } j = 1, 2, \dots, K,$$

where $\mathbf{x}_j^{(i)}$ is the j th (n/K) -element sub-vector of the vector $\mathbf{x}^{(i)}$.

In the result of above in every step we must solve K equation systems of n/K size each in the following form:

$$\mathbf{D}_{jj}\mathbf{x}_j^{(i+1)} = \mathbf{z}_j^{(i+1)} \quad \text{for } j = 1, 2, \dots, K, \quad (8)$$

where

$$\mathbf{z}_j^{(i+1)} = \left(\sum_{l=1}^{i-1} \mathbf{L}_{jl}\mathbf{x}_l^{(i+1)} + \sum_{l=j+1}^K \mathbf{U}_{jl}\mathbf{x}_l^{(i)} \right) \quad \text{for } j = 1, 2, \dots, K.$$

We can apply different direct and iterative methods to solve equation (8). For every block we make factorization only once before iteration, in loop we change only right equation sides. If \mathbf{D}_{ii} matrix has special structure, for example is a diagonal, upper triangular, lower triangular or tri-diagonal matrix—then LU factorization is very simple to get and iterative-block method becomes very attractive. If \mathbf{D}_{ii} matrices do not have any of mentioned structures, then it can appear that block solving demands iterative methods. Then, we have inside iterative method with cardinal iterative method. It demands selecting of proper method for inside method to be convergent and initial vector to be chosen for the given method. There is small number of iterations demanded to obtain convergence for small number of blocks (sub-matrices are big).

5 Numerical experiment

Algorithms implementation was done in C language. Data structures to save matrices \mathbf{Q}^T , \mathbf{W} , \mathbf{Z} , \mathbf{L} , \mathbf{U} were full two-dimensional arrays situated in RAM. Numerical experiment was conducted on PC computer with 1GB RAM, Pentium IV 2.80 GHz processor. Algorithms were tested in Linux environment and compiler gcc with -O3 option was used in compilation. Tests were conducted for sparse matrices generated by authors. Generated matrices described abstract queuing models and they had all the attributes of transition rate matrices \mathbf{Q}^T . Table 1 shows information about test matrices for which IWZ and ILU incomplete factorizations were applied—those matrices were not symmetric at all. Where n means columns number and nz means non-zeros number.

Tested matrices differed with average non-zero elements number in column. Two matrices had more than 10 elements in column (group I) and the other two had about four elements in column (group II). It essentially affects calculations accuracy and iterative methods convergence.

Table 1. Test matrices attributes

group	ID	n	nz	average non-zeros per column (nz/n)
I	1	1000	12678	12.68
I	2	4000	42041	10.51
II	3	1500	5873	3.92
II	4	4000	16946	4.24

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

Table 2. Comparing of algorithms calculation time (GS, GSWZ, GSLU) and an impact of algorithms on result accuracy

ID(n)	time and accuracy	GS	GSWZ		GSLU	
			$K = 2$	$K = 20$	$K = 2$	$K = 20$
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

Applying block GS (in variants GSLU and GSWZ) comparing to classic GS gave the following observation:

- For all the matrices—dividing into small number of blocks influences faster method’s convergence. But dividing into great number of blocks makes accuracy to be obtained at the same number of steps like for usual GS.
- Time of algorithm calculating depends only upon blocks number independently on matrices groups. The least blocks number is the longer time of calculations—what is normal for great matrices to be solved with accurate methods and cost depends on size. Dividing into great number of blocks shortens calculations time. The more blocks are the solving time becomes similar to usual Gauss-Seidel method time solving.
- Calculations accuracy does not depend on factorization: WZ or LU. But factorization influences time of calculation. Dividing into great number of blocks, we get 1.5 time faster algorithm GSWZ than GSLU.

Incomplete factorizations IWZ and ILU were applied for matrix \mathbf{Q}^T as preconditioning. Iterative Gauss-Seidel method was applied for obtained equation system (6) and (7) or for block Gauss-Seidel. Preconditioning can be used for all iterative, projective and other methods.

Table 3. Comparing of time of algorithms (GS, IWZ+GS, ILU+GS, GSWZ, GSLU) and its accuracy

ID(n)	algorithm	time	$i = 5$	$i = 10$	$i = 20$
1(1000)	GS	0,18	e-04	e-08	e-16
	IWZ+GS	0,17	e-08	e-12	e-18
	ILU+GS	0,17	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
2(4000)	GS	2,14	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
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
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

Table 3 shows the comparison of the methods: traditional Gauss-Seidel algorithm (denoted: GS), preconditioned GS—both with IWZ (denoted: IWZ+GS)

and with ILU (denoted: ILU+GS), and block GS, where blocks were solved with WZ (denoted: GSWZ(K)) and with LU (denoted: GSLU(K)). Here, the number of blocks is K^2 . There was chosen $K = 2$, because of the best accuracy. The time was compared, as well as convergence after 5, 10 and 20 iterations.

The results observed:

- The convergence of matrices from group II (more sparse matrices) is weak so they need more accuracy followed by iteration number reduction or applying preconditioning or blocking.
- Applying both preconditioning and blocking improves convergence of GS iterative methods.
- Preconditioning accelerates iterative methods convergence—especially using incomplete IWZ factorization for usual GS regarding both calculations time and accuracy.

6 Conclusions

In above article the authors compared two techniques which can affect calculations accuracy: preconditioning and blocking in special linear equation systems solving. Matrix coefficients of those systems describe transition rates from one state to another state described by Markov processes. Numerical experiment results show that it is better to apply usual Gauss-Seidel preconditioning for these systems than blocking. It also should be noted that if we have IWZ and ILU to chose it is better to chose IWZ—having shorter calculations time and more accuracy. Moreover it was observed that convergence of matrices from group II is weak—so they need special techniques applying to improve accuracy what is followed by reducing of iterations number and using preconditioning or blocking: teats result in applying preconditioning -as the best method. Matrix should be divided into less blocks number to improve accuracy what affects calculation time (longer one). In next articles we plan to join block methods and preconditioning as well as conducting of calculations time.

Acknowledgements

This work was partially supported by Marie Curie-Sklodowska University in Lublin within the project *Niekompletny rozkład WZ jako uwarunkowanie wstępne (ang. preconditioning) stosowane do znajdowania wektora prawdopodobieństwa stanów sieci całkowicie optycznych modelowanych łańcuchami Markowa*.

This work was also partially supported within the project *Metody i modele dla kontroli zatłoczenia i oceny efektywności mechanizmów jakości usług w Internecie następnej generacji* (N517 025 31/2997).

References

1. Benzi M., Ucar B.: *Block Triangular preconditioners for M-matrices and Markov chains*, [To appear in Electronic Transactions on Numerical Analysis, 26 (2007).]

2. J. Bylina: *Distributed solving of Markov chains for computer network models*, *Anales UMCS Informatica* 1 (2003), pp. 15–20.
3. M. M. Chawla, R. R. Khazal: *A new WZ factorization for parallel solution of tridiagonal systems*, *Int. J. Comput. Math.* 80 (2003), no. 1, 123–131.
4. I. S. Duff: *Combining direct and iterative methods for the solution of large systems in different application areas*, Technical Report RAL-TR-2004-033.
5. D. J. Evans, M. Barulli: *BSP linear solver for dense matrices*, *Parallel Computing* 24 (1998), pp. 777–795.
6. D. J. Evans, M. Hatzopoulos: *The parallel solution of linear system*, *Int. J. Comp. Math.* 7 (1979), pp. 227–238.
7. R. E. Funderlic, C. D. Meyer: *Sensitivity of the stationary distribution vector for an ergodic Markov chain*, *Linear Algebra Appl.*, 76, 1986, pp. 1–17.
8. R. E. Funderlic, R. J. Plemmons: *Updating LU factorizations for computing stationary distributions*, *SIAM J. Alg. Disc. Meth.*, 7, 1986, pp. 30–42.
9. G. H. Golub, C. D. Meyer: *Using the QR factorization and group inversion to compute, differentiate and estimate the sensitivity of stationary distributions for Markov chains*, *SIAM J. Alg. Disc. Meth.*, 7, 1986, pp. 273–281.
10. W. J. Harrod, R. J. Plemmons: *Comparisons of some direct methods for computing stationary distributions of Markov chains* *SIAM J. Sci. Comput.*, 5, 1984, pp. 453–469.
11. M. Haviv: *Aggregation/disaggregation methods for computing the stationary distribution of a Markov chain*, *SIAM J. Numer. Anal.*, 24, 1987, pp. 952–966.
12. A. Jennings, W. J. Stewart: *Simultaneous iteration for partial eigensolution of real matrices*, *J. IMA*, 15, 1975, pp. 351–361.
13. A. Jennings, W. J. Stewart: *A simultaneous iteration algorithm for real matrices*, *ACM Trans. of Math. Software*, 7, 1981, pp. 184–198.
14. Y. Saad, M. H. Schultz: *GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems*, *SIAM Journal of Scientific and Statistical Computing*, 7, 1986, pp. 856–869.
15. P. J. Schweitzer, K. W. Kindle: *An iterative aggregation-disaggregation algorithm for solving linear systems*, *Applied Math. and Comp.*, 1986, 18, pp. 313–353.
16. W. Stewart: *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Chichester, West Sussex 1994.
17. P. Yalamov, D. J. Evans: *The WZ matrix factorization method*, *Parallel Computing* 21 (1995), pp. 1111–1120.