



Department of Mathematics
Faculty of Mechanical Engineering
Slovak University of Technology in Bratislava

3rd International Conference
APLIMAT 2004

SOLVING MARKOV CHAINS WITH THE WZ FACTORIZATION FOR MODELLING NETWORKS

BYLINA Jarosław (PL), BYLINA Beata (PL)

Abstract. In the paper we want to pay attention to the WZ factorization – a manner for solving linear equations which appear in modelling of computer and communication networks with Markov chains. We present two approaches to that factorization – specific ones for solving Markov chains. We present results of the numerical experiments for these two algorithms. Namely, the probability vector was computed in those two ways for a transition matrix generated with suitable tools (for the behaviour of a simple optical switch) and accuracy of both manners was compared.

1 Introduction

A queuing network model represents a computer network as a system of service stations with customers moving among them. The service stations are resources of the given system, The customers (served at the service stations) are jobs for the system or information packages.

Probabilistic methods are the most useful ones of the mathematical methods to describe queuing models. Every probabilistic model can be approximated (with an arbitrary accuracy) with a Markov model (where the model state in a time moment depends only on the state in the directly previous moment).

A continuous time Markov chain can be described with a single matrix $\mathbf{Q} = (q_{ij})_{1 \leq i, j \leq n}$ called the transition rate matrix (or the infinitesimal generator) given by:

$$q_{ij}(t) = \lim_{\Delta t \rightarrow 0} \frac{p_{ij}(t, t + \Delta t)}{\Delta t} \quad \text{for } i \neq j,$$
$$q_{ii}(t) = - \sum_{j \neq i} q_{ij}(t)$$

where $p_{ij}(t_1, t_2)$ is the probability that when the model is in the state i in the moment t_1 the transition occurs to the state j before the moment t_2 .

We are to find $\pi(t)$ – the (horizontal) vector of the probabilities $\pi_i(t)$ that the system is in the state i at the time t .

If we are given a steady-state (i.e. there exists $\pi = \lim_{t \rightarrow \infty} \pi(t)$) homogeneous (i.e. \mathbf{Q} didn't depends on t) probability distribution (what we can often assume) to find π , we are to solve a linear system with some constraints:

$$\pi \mathbf{Q} = 0, \quad \pi \geq 0, \quad \pi \mathbf{e} = 1 \quad (1)$$

where $\mathbf{e} = (1, 1, \dots, 1)^T$

The matrix \mathbf{Q} is a square $n \times n$ matrix (usually a huge one), of the rank $(n - 1)$ (in interesting for us, homogenous cases), with the dominant diagonal. It is singular, so we need appropriate approaches to the system (1).

There are indirect methods that are widely used – like iterative and projection methods. However, sometimes we need a direct method (when we need the most accurate results, for example). The most popular of the direct methods are the LU factorization method and its relatives.

In this article we want to present another direct method for solving (1), namely the WZ factorization method which can be faster for matrices with the dominant diagonal – as was shown in [8]. In Section 2 we present the WZ factorization and in Section 3 we present a sequential algorithm for solving linear systems with the WZ factorization. In section 4 we describe two approaches to solving (1) – both of them use the WZ factorization. Section 5 contains numerical algorithms described in Section 4 and details of implementation of these algorithms. In Section 6 we present the conclusion.

2 The WZ Factorization

In this section we describe shortly the WZ factorization and the method for solving a linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad \text{where} \quad \mathbf{A} \in \mathcal{R}^{n \times n}, \quad \mathbf{b} \in \mathcal{R}^n$$

with the WZ factorization.

The WZ factorization is described in [3, 4]: $\mathbf{A} = \mathbf{W}\mathbf{Z}$, where matrices \mathbf{W} and \mathbf{Z} consist of the following columns \mathbf{w}_i and rows \mathbf{z}_i^T respectively:

$$\begin{aligned} \mathbf{w}_i &= (\underbrace{0, \dots, 0}_i, 1, w_{i+1,i}, \dots, w_{n-i,i}, 0, \dots, 0)^T & i = 1, \dots, m, \\ \mathbf{w}_i &= (\underbrace{0, \dots, 0}_i, 1, 0, \dots, 0)^T & i = p, q, \\ \mathbf{w}_i &= (\underbrace{0, \dots, 0}_{n-i+1}, w_{n-i+2,i}, \dots, w_{i-1,i}, 1, 0, \dots, 0)^T & i = q + 1, \dots, n, \\ \mathbf{z}_i^T &= (\underbrace{0, \dots, 0}_{i-1}, z_{ii}, \dots, z_{i,n-i+1}, 0, \dots, 0) & i = 1, \dots, p, \\ \mathbf{z}_i^T &= (\underbrace{0, \dots, 0}_{n-1}, z_{i,n-i+1}, \dots, z_{ii}, 0, \dots, 0) & i = p + 1, \dots, n, \end{aligned} \quad (2)$$

where

$$m = \lfloor (n - 1)/2 \rfloor, \quad p = \lfloor (n + 1)/2 \rfloor, \quad q = \lceil (n + 1)/2 \rceil.$$

After factorization we can solve two linear systems:

$$\mathbf{W}\mathbf{c} = \mathbf{b}, \quad (3)$$

$$\mathbf{Z}\mathbf{x} = \mathbf{c}, \quad (4)$$

(where \mathbf{c} is an auxiliary vector) instead of one (1).

3 Sequential Algorithm for WZ Factorization

The WZ method algorithm for solving linear systems consist of two parts: reduction of the matrix \mathbf{A} (and the vector \mathbf{b}) to the matrix \mathbf{Z} (and the vector \mathbf{c}) and next solving equation (4). The first part consists in partial zeroing of columns of the matrix \mathbf{A} . In the first step we zero elements from the 2nd to $n - 1$ st in the 1st and n th column. Next, we update the matrix \mathbf{A} and the vector \mathbf{b} . Formally we can write this (after [3, 8], for $k = 0, \dots, m$):

Step $k.1$. We compute w_{i1} and w_{in} (for $i = 2, \dots, n - 1$) from the linear system:

$$\begin{cases} a_{1+k,1+k}w_{i,1+k} + a_{n-k,1+k}w_{i,n-k} = -a_{i,1+k}, \\ a_{1+k,n-k}w_{i,1+k} + a_{n-k,n-k}w_{i,n-k} = -a_{i,n-k}. \end{cases}$$

Step $k.2$. We compute

(for $i, j = 2 + k, \dots, n - 1 - k$):

$$\begin{aligned} a_{i,j}^{(k+1)} &= a_{i,j}^{(k)} + w_{i,1+k}a_{1+k,j}^{(k)} + w_{i,n-k}a_{n-k,j}^{(k)}, \\ b_i^{(k+1)} &= b_i^{(k)} + w_{i,1+k}b_{1+k}^{(k)} + w_{i,n-k}b_{n-k}^{(k)}. \end{aligned}$$

After $m + 1$ such steps we get the matrix $\mathbf{Z} = \mathbf{A}^{(m+1)}$, (as defined in (2)) and the vector $\mathbf{c} = \mathbf{b}^{(m+1)}$.

We get

$$\mathbf{W}^{(m)} \cdot \dots \cdot \mathbf{W}^{(0)} \cdot \mathbf{A} = \mathbf{Z}$$

so

$$\mathbf{A} = \{\mathbf{W}^{(m)}\}^{-1} \cdot \dots \cdot \{\mathbf{W}^{(0)}\}^{-1} \cdot \mathbf{Z} = \mathbf{WZ}.$$

The second part of the method is to solve a linear system (4). This part consists in solving a linear system with two unknown quantities x_p and x_q and next updating the vector \mathbf{b} . Formally we can describe this for the k th step ($k = 0, \dots, m$):

Step $k.1$. We find x_{p-k} and x_{q+k} from the system:

$$\begin{cases} z_{p-k,p-k}x_{p-k} + z_{p-k,q+k}x_{q+k} = c_{p-k}^{(k)}, \\ z_{q+k,p-k}x_{p-k} + z_{q+k,q+k}x_{q+k} = c_{q+k}^{(k)}. \end{cases} \quad (5)$$

Step $k.2$. We compute (for $i = 1 + k, \dots, p - 1, q + 1, \dots, n - k$):

$$c_i^{(k+1)} = c_i^{(k)} - z_{i,p-k}x_{p-k} - z_{i,q+k}x_{q+k}.$$

For the odd n equation system (5) consists in the first step of one equation.

4 Algorithms

As we noted above (Section 1) we are interested in finding the vector π from (1).

In our case we need to solve a homogenous linear system with n equations and n unknowns. Such a system has a non-zero solution if and only if the coefficient matrix is singular. The matrix \mathbf{Q} has a zero eigenvalue ([7]) so it is singular. Let us assign $\pi = \mathbf{x}^T$ and transpose (1) to get:

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

We want to present two approaches to solving such a linear system. For both of them we assume that the Markov chain represented by \mathbf{Q} is irreducible (so \mathbf{Q} is of rank $(n - 1)$).

4.1 Zero determinant

There exist matrices \mathbf{W} and \mathbf{Z} described in Section 2, meeting an equity $\mathbf{Q}^T = \mathbf{WZ}$ and \mathbf{W} is not singular (see [3]). We are to solve an equation $\mathbf{WZx} = 0$, with $\mathbf{e}^T \mathbf{x} = 1$. To solve it we can solve two linear systems: $\mathbf{Wy} = 0$ and $\mathbf{Zx} = \mathbf{y}$. The matrix \mathbf{W} is not singular (see (2)) so the only solution of $\mathbf{Wy} = 0$ is $\mathbf{y} = 0$. We have only one system $\mathbf{Zx} = 0$ to solve.

When n is an even number that system has a following form:

$$\begin{bmatrix} z_{11} & \cdot & \cdot & \cdot & \cdot & z_{1n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \mathbf{0} & & z_{pp} & z_{pq} & \mathbf{0} & \\ \cdot & & z_{qp} & z_{qq} & \cdot & \\ z_{n1} & \cdot & \cdot & \cdot & \cdot & z_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_p \\ x_q \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (6)$$

To solve the system (6) we do as described in Section 3 – first we solve a two-equation linear system:

$$\begin{cases} z_{pp}x_p + z_{pq}x_q = 0, \\ z_{qp}x_p + z_{qq}x_q = 0. \end{cases} \quad (7)$$

But $z_{pp}z_{qq} - z_{pq}z_{qp} = 0$ (it is a 2×2 determinant of the last step of elimination for the matrix of rank $(n - 1)$ with dominant diagonal). Moreover, of course $0 \cdot z_{qq} - z_{pq} \cdot 0 = 0$ and $z_{pp} \cdot 0 - 0 \cdot z_{qp} = 0$ so our system (7) has infinite number of solutions. We choose an arbitrary non-zero real number ξ and set $x_p = \xi$. Now we can compute other elements of the vector \mathbf{x} .

After computing the whole vector \mathbf{x} we normalize it to meet the equity $\mathbf{e}^T \mathbf{x} = 1$.

In the case of an odd number n (when $p = q$) the only difference is that the system $\mathbf{Zx} = 0$ has the form:

$$\begin{bmatrix} z_{11} & \cdot & \cdot & \cdot & z_{1n} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \mathbf{0} & & z_{pp} & \mathbf{0} & \\ \cdot & \cdot & \cdot & \cdot & \\ z_{n1} & \cdot & \cdot & \cdot & z_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_p \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

There holds $z_{pp} = 0$ (it is a 1×1 determinant of the last step of elimination for the matrix of rank $(n - 1)$ with dominant diagonal) so the equation $z_{pp}x_p = 0$ has infinite number of solutions. We can set $x_p = \xi$ ($\xi \neq 0$) and find other elements of the vector \mathbf{x} .

After computing the whole vector \mathbf{x} we normalize it to meet the equity $\mathbf{e}^T \mathbf{x} = 1$.

4.2 Replacing an equation

An alternative approach to solving the system (1) is to replace an arbitrary equation of that system with the normalization equation

$$\mathbf{e}^T \mathbf{x} = 1.$$

Let $\tilde{\mathbf{Q}}_p$ be the matrix \mathbf{Q} with the p th column replaced with the vector \mathbf{e} . Our modified system can be written:

$$\tilde{\mathbf{Q}}_p^T \mathbf{x} = \mathbf{e}_p, \quad \text{where} \quad \mathbf{e}_p = (e_i)_{1 \leq i \leq n}, \quad e_i = \delta_{ip}$$

Let $\tilde{\mathbf{Q}}_p^T = \tilde{\mathbf{W}}\tilde{\mathbf{Z}}$. Setting $\tilde{\mathbf{Z}}\mathbf{x} = \mathbf{y}$ in the system $\tilde{\mathbf{W}}\tilde{\mathbf{Z}}\mathbf{x} = \mathbf{e}_p$ we get $\tilde{\mathbf{W}}\mathbf{y} = \mathbf{e}_p$ from which it is obvious that $\mathbf{y} = \mathbf{e}_p$. So now we are to solve the system $\tilde{\mathbf{Z}}\mathbf{x} = \mathbf{e}_p$.

Obviously, it is not necessary to replace the p th equation, but it is optimal from the point of view of the time complexity (in this case we know the solution of the equation $\tilde{\mathbf{W}}\mathbf{y} = \mathbf{e}_p$ in advance).

In this algorithm it is not important whether n is odd or even.

This approach is likely to yield a less accurate result than the first one. When we compute x_p and x_q they will be contaminated with the round-off errors from all of the previous elimination steps. Moreover, this will propagate throughout the next backsubstitution steps. In the first method we use an uncontaminated constant ξ which disappears after normalization.

5 The implementation and the numerical experiments

Both the algorithms were implemented for single precision numbers with the use of the language *C* [5]. Programs were compiled with the compiler *gcc* with the use of the maximal optimization option (-O3). The programs were tested under the *Linux* operating system on the computer with the processor Pentium-4 2.8 GHz and with 1 GB RAM.

Data for our tests were generated with the library *OLYMP* [6]. They are taken from the following example model:

Information packages arrives to a buffer (of an optical switch) of capacity of $N = 250$ blocks. The packages are of different sizes (sizes are integer, from one block up to 20 blocks). When the buffer is filled the bigger (optical) package is formed and sent. The buffer is then emptied. We don't want to divide information from one package between two optical packages so when an arriving package is too big to fit into the buffer we send an uncompleted optical package and package just received starts a new optical package. An uncompleted optical package is also sent when a given timeout $T = 1$ is over. Arriving packages stream has a Poisson distribution with $\lambda = 1$ and the probability that the received package has i blocks is $p_i = 0.05$. In our model T is approximated with an Erlang distribution consisting of 10 phases of exponential distribution with $\mu = 10/T = 10$, because we want to preserve a Markovian form of this model.

In direct methods of solving equations elimination of one non-zero element in the reaction phase can cause more non-zero elements to appear where there were zeros. It forces us to think about storage schemes for sparse matrices (like the one in our example). However, in

our example the matrix – despite of its sparsity – was rather small so we decided to store it in a traditional – dense – form. That turned out to be the best way because of the access and insert time.

The performances of the algorithms were similar (396 s for “Zero determinant” and 405 s for “Replacing an equation”), but the difference in accuracy is significant (residual norm equal to $1.36509 \cdot 10^{-7}$ for “Zero determinant” and $5.63048 \cdot 10^{-7}$ for “Replacing an equation”).

6 Conclusion

Presented above algorithms for solving linear systems (1) which appear during Markovian modelling of computer networks are direct methods. They are alternative to iteration and projection methods and also to the classic LU factorization.

We showed that our algorithms are rather fast for our problem solved on a one-processor machine. The difference is in computation accuracy.

Presented algorithms can be easily parallelized and vectorized [1, 2] what can allow us to solve bigger and more complicated problems (having more complicated mechanisms for packages flow, more complex arriving/servicing distributions etc.) with the Markov chains.

References

- [1] Bylina, B.: Solving linear systems with vectorized WZ factorization, In: *Annales UMCS Informatica* **1**, pp. 5–13, 2003.
- [2] Bylina, B., Bylina J.: Efficient matrix-vector algorithm for solving linear systems by WZ factorization, In: *Lecture Notes on Computer Science – PPAM Proceedings*, 2003 (to appear).
- [3] Chandra Sekhara Rao, S.: Existence and uniqueness of WZ factorization, In: *Parallel Computing* **23**, pp. 1129–1139, 1997.
- [4] Evans, D.J., Hatzopoulos, M.: The parallel solution of linear system, In: *Int. J. Comp. Math.* **7**, pp. 227–238, 1979.
- [5] Kernighan, B.W., Ritchie, D.M.: *The C Programming Language, Second Edition*, Prentice Hall Inc., 1988.
- [6] Pecka, P.: *Obiektowo zorientowany system wielowątkowy do modelowania stanów nieustalonych w sieci komputerowej metodą łańcuchów Markowa*, Ph.D. thesis, Gliwice, 2002 (in Polish).
- [7] Stewart, W.: *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Chichester, West Sussex, 1994.
- [8] Yalamov, P., Evans D.J.: The WZ matrix factorization method, In: *Parallel Computing* **21**, pp. 1111–1120, 1995.

Contact address

Mgr. Beata Bylina, Mgr. Jarosław Bylina, Department of Computer Science, Institute of Mathematics, Maria Curie-Skłodowska University, Pl. M. Curie-Skłodowskiej 1, 20-031 Lublin, Poland

e-mail: beatas@hektor.umcs.lublin.pl, jmbylina@hektor.umcs.lublin.pl