Annales UMCS Informatica AI 1 (2003) 5-13

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

Solving linear systems with vectorized WZ factorization

Beata Bylina*

Departament of Computer Science, Institute of Mathematics, Maria Curie-Skłodowska University
Pl. M. Curie-Skłodowskiej 1, 20-031 Lublin, Poland

Abstract

In the paper we present a vectorized algorithm for WZ factorization of a matrix which was implemented with the BLAS1 library. We present the results of numerical experiments which show that vectorization accelerates the sequential WZ factorization. Next, we parallelized both algorithms for a two-processor shared memory machine using the OpenMP standard. We present performances of these four algorithms on a two-Pentium III machine with a Linux system (the parallelized sequential algorithm is better than the normal sequential one, but the parallelized vectorized algorithm is very similar in its performance to the non-parallelized vectorized one).

1. Introduction

Solution of the linear systems

$$Ax = b, \qquad A \in \Re^{n \times n},\ b \in \Re^{n} \tag{1}$$

is an important problem in scientific and engineering computations. One of the methods to solve a dense linear system is its WZ factorization. Matrix A is factorized to a product of matrices W and Z (which are described in Section 2). Such a factorization exists for a nonsingular matrix as shown in [1]. In Section 3 we present (according to [2-4]) the idea of solution of the linear systems (1) with WZ factorization of the matrix and in Section 4 we describe the algorithm of WZ factorization using the vector notation. We present algorithm for matrices, which can be factorized without pivoting, that is for symmetric positive definite and strictly diagonally dominant ones (as proved in [1]). Next, in Section 5, we describe the results of our experiments for the sequential and vectorized WZ factorization algorithms. Next, both the algorithms are partially parallelized (using OpenMP standard) and run on a 2-processor shared-memory machine with the Linux-based operating system. The results of the experiments are described and conclusions are presented in Section 5.

* *E-mail address*: beatas@golem.umcs.lublin.pl

2. WZ factorization

In this Section we describe the method of a matrix WZ factorization and solving linear systems (1). WZ factorization is described in [1, 2, 5, 6]. Let's assume A is a nonsingular $n\times n$ matrix. $A=WZ$, where the matrices W and Z consist of following columns w_i and rows z_i respectively:

$$\begin{aligned}
w_i &= (0\,\underbrace{\ldots}_{i}\,0\,1\,w_{i+1,i}\,\ldots\,w_{n-i,i}\,0\,\ldots\,0)^T, i=1,\ldots,m,\\
w_i &= (0\,\underbrace{\ldots}_{i}\,0\,1\,0\,\ldots\,0)^T, i=p,q,\\
w_i &= (0\,\underbrace{\ldots}_{n-i+1}\,0\,w_{n-i+2,i}\,\ldots\,w_{i-1,i}\,1\,0\,\ldots\,0)^T, i=q+1,\ldots,n,\\
z_i &= (0\,\underbrace{\ldots}_{i-1}\,0\,z_{ii}\,\ldots\,z_{i,n-i+1}\,0\,\ldots\,0), i=1,\ldots,p,\\
z_i &= (0\,\ldots\,0\,z_{i,n-i+1}\,\ldots\,z_{ii}\,0\,\ldots\,0), i=p+1,\ldots,n,
\end{aligned} \tag{2}$$

where

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

For example, for n=5 and n=6 we have:

$$W=\begin{bmatrix} 1 & 0 & 0 & 0 & 0\\ w_{21} & 1 & 0 & 0 & w_{25}\\ w_{31} & w_{32} & 1 & w_{34} & w_{35}\\ w_{41} & 0 & 0 & 1 & w_{45}\\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},\quad Z=\begin{bmatrix} z_{11} & z_{12} & z_{13} & z_{14} & z_{15}\\ 0 & z_{22} & z_{23} & z_{24} & 0\\ 0 & 0 & z_{33} & 0 & 0\\ 0 & z_{42} & z_{43} & z_{44} & 0\\ z_{51} & z_{52} & z_{53} & z_{54} & z_{55} \end{bmatrix},\quad n=5,$$

$$W=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0\\ w_{21} & 1 & 0 & 0 & 0 & w_{26}\\ w_{31} & w_{32} & 1 & 0 & w_{35} & w_{36}\\ w_{41} & w_{42} & 0 & 1 & w_{45} & w_{46}\\ w_{52} & 0 & 0 & 0 & 1 & w_{56}\\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},\quad Z=\begin{bmatrix} z_{11} & z_{12} & z_{13} & z_{14} & z_{15} & z_{16}\\ 0 & z_{22} & z_{23} & z_{24} & z_{25} & 0\\ 0 & 0 & z_{33} & z_{34} & 0 & 0\\ 0 & 0 & z_{43} & z_{44} & 0 & 0\\ 0 & z_{52} & z_{53} & z_{54} & z_{55} & 0\\ z_{61} & z_{62} & z_{63} & z_{64} & z_{65} & z_{66} \end{bmatrix},\quad n=6.$$

After factorization we can solve the two linear systems:

$$Wc=b,$$

$$Zx=c,$$

instead of one (1).

3. Sequential algorithm of the WZ factorization

The WZ method algorithm of solving linear systems consists of two parts: reduction of the matrix A (and the vector b) to the matrix Z (and the vector c) and next solving equation $Zx=c$.

The first part consists in partial zeroing of columns of the matrix A. In the first step we zero the elements from the 2nd to n–1st in the first and nth column. Next we update the matrix A and the vector b. Formally we can write this (after [5]):

Step 1.1. We compute $\overline{w}_{i1}$ and $\overline{w}_{in}$ from the linear system:

$$\begin{cases} a_{11}\overline{w}_{i1} + a_{n1}\overline{w}_{in} = -a_{i1} \\ a_{1n}\overline{w}_{i1} + a_{nn}\overline{w}_{in} = -a_{in}, \quad \text{for} \quad i = 2, \mathbf{K}, n-1 \end{cases}$$

and we get the matrix:

$$W^{(1)} = \begin{bmatrix} 1 & & & 0 \\ \overline{w}_{21} & 1 & & \overline{w}_{2n} \\ \mathbb{M} & & \mathbf{O} & \mathbb{M} \\ \overline{w}_{n-1,1} & & 1 & \overline{w}_{n-1,n} \\ 0 & & & 1 \end{bmatrix}.$$

Step 1.2. We compute

$$A^{(1)} = W^{(1)}A, \quad b^{(1)} = W^{(1)}b.$$

After the first step we get the linear system $A^{(1)}x = b^{(1)}$ where

$$A^{(1)} = \begin{bmatrix} a_{11} & a_{12} & \mathbf{L} & a_{1,n-1} & a_{1n} \\ 0 & a_{22}^{(1)} & \mathbf{L} & a_{2,n-1}^{(1)} & 0 \\ \mathbb{M} & \mathbb{M} & & \mathbb{M} & \mathbb{M} \\ 0 & a_{n-1,2}^{(1)} & \mathbf{L} & a_{n-1,n-1}^{(1)} & 0 \\ a_{n1} & a_{n2} & \mathbf{L} & a_{n,n-1} & a_{nn} \end{bmatrix}, \quad b^{(1)} = \begin{bmatrix} b_1 \\ b^{(1)} \\ \mathbb{M} \\ b_{n-1}^{(1)} \\ b_n \end{bmatrix},$$

and

$$\begin{aligned} a_{ij}^{(1)} &= a_{ij} + \overline{w}_{i1}a_{1j} + \overline{w}_{in}a_{nj}, \quad j = 2, \mathbf{K}, n-1, \quad i = 2, \mathbf{K}, n-1, \\ b_i^{(1)} &= b_i + \overline{w}_{i1}b_1 + \overline{w}_{in}b_n, \quad i = 1, \mathbf{K}, n-1. \end{aligned} \tag{3}$$

Similarly, we carry out the second step (and next steps) which consist in the same operations as before, but only on the submatrices of $A^{(1)}$ obtained by deleting the first and the last rows and columns of the matrix $A^{(1)}$.

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

$$W^{(m)}\,\mathbf{K}\,W^{(1)}A = Z$$

so

$$A = \{W^{(1)}\}^{-1}\,\mathbf{K}\,\{W^{(m)}\}^{-1}Z = WZ.$$

The second part of the method is to solve the linear system $Zx = b^{(m)}$. This part consists in solving a linear system with two unknown quantities x_p and x_q and next updating the vector b. Formally we can describe this for the first step:

Step 1.1. We find x_p and x_q from the system:

$$\begin{cases} z_{pp}x_p + z_{pq}x_q = b_p^{(m)} \\ z_{qp}x_p + z_{qq}x_q = b_q^{(m)}. \end{cases} \tag{4}$$

Step 1.2. We compute:

$$c_i^{(1)} = c_i - z_{ip}x_p - z_{iq}x_q, \quad i = 1, \mathbf{K}, p-1, q+1, \mathbf{K}, n. \tag{5}$$

For the odd n equation system (4) consists (in the last step) of one equation. Similarly, we make the next steps for the inner two-equation systems. There are $m+1$ such steps.

In the remainder of our paper we use the name AWZ to denote our sequential WZ algorithm (described above).

4. Vectorized WZ factorization algorithm

In this Section we present a new WZ factorization algorithm. We describe the algorithm without pivoting, working only for matrices for which the WZ factorization is executable. We use the MATLAB notation [7] for describing algorithm (for matrices of even sizes).

```
% elimination loop – reduction steps for the matrix A to Z
for k = 0:m-1
  k2 = n-k-1
  det = A(k, k)*A(k2, k2) – A(k2, k)*A(k, k2)
  for i = k+1:k2-1
% computation of coefficients
    wk1 = (A(k2, k)*A(i, k2) – A(k2, k2)*A(i, k))/det
    wk2 = (A(k, k2)*A(i, k) – A(k, k)*A(i, k2))/det
% updating the matrix A
    A(i,k+1:k2-1)
= A(i,k+1:k2-1)+wk1*A(k,k+1:k2-1)+wk2*A(k2,k+1:k2-1)
% updating the vector b
  b(i) = b(i) + wk1*b(k) + wk2*b(k2)
% finding the vector x
for j = m:0
% solving a 2x2 system
  j2 = n – j + 1
  det = A(j, j)*A(j2, j2) – A(j, j2)*A(j2, j)
```

```
  x(j) = (b(j)*A(j2, j2) – b(j2)*A(j, j2))/det
  x(j2) = (b(j2)*A(j, j) – b(j)*A(j2, j))/det
```
% updating $b(1{:}j{-}1)$ ('upper part')
```
  b(1:j–1)
= b(1:j-1)–x(j)*A(1:j–1,j) – x(j2)* A(1:j-1,j2)
```
% updating $b(n{-}j{+}2{:}n)$ ('lower part')
```
  b(n-j+2:n)
= b(n–j+2:n)–x(j)*A(n–j+2:n,j)–x(j2)*A(n–j+2:n,j2)
```

In our paper we use the name VAWZ to denote the vectorized WZ algorithm.

5. Implementation and numerical experiments

The algorithms AWZ and VAWZ were implemented with the language C [8] using double precision. The programs were compiled with Intel C Compiler for Linux (icc) [9], additionally program VAWZ was linked with the library BLAS (Basic Linear Algebra Subprograms) [10]. BLAS is the collection of subprograms helping with linear algebra computing. In BLAS we can use real and complex numbers in single and double precision. We use BLAS1 (Basic Linear Algebra Subprograms level 1) that contains vector-vector operations. To implement the algorithm VAWZ we use BLAS1 functions of the type $x \leftarrow x + ay$ (_daxpy).

The algorithms were tested on a two-processor (Pentium III 733 MHz) machine working under the Linux operating system. Both the algorithms were run for matrices for which WZ factorization is possible. Here are the results.

1. The speed of the algorithm VAWZ is statistically 92% higher than the speed of the algorithm AWZ. For the matrices of size greater the 1000 VAWZ is 53% greater than the speed of the algorithm AWZ. The speed decreases with the growth of the size for both the algorithms but falls more and more slowly to the certain level. We can view that the speed of the algorithm AWZ stabilizes for the size of 1300 at about 38 Mflops and the speed of the algorithm VAWZ stabilizes for the same size at about 65 Mflops. We can see the results in the Figure 1.

Next we parallelized both the algorithms using OpenMP directives. OpenMP [10] is a standardized set of mechanisms (directives, functions etc.) for creating parallel programs for shared memory machines. OpenMP is supported by many producers of such machines/architectures – so is by Intel.

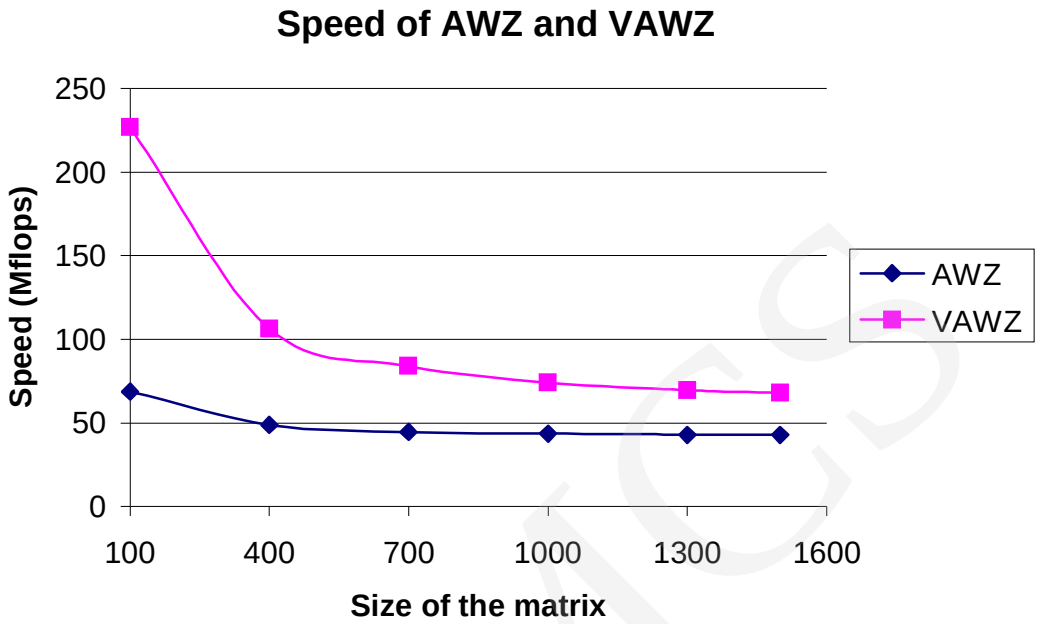

Fig. 1. Speed of AWZ and VAWZ

Our parallel programs were compiled with Intel C Compiler (icc). A parallel program compiled that way (with icc using standard OpenMP) starts as a single thread running as such to the first parallel construction. At that moment the main thread is forked and different threads are run by different processors.

In our algorithms we parallelized the following elements:

1. In the first part (which computes the matrix Z and the vector c) we update parallelly rows of the matrix A and elements of the vector b.
2. In the second part (which finds the solution vector x) we update parallelly upper and lower parts of the vector b.

In the paper we use the name PAWZ to denote the parallelized AWZ and PVAWZ to denote the parallelized VAWZ.

Both the algorithms were tested for matrices for which WZ factorization is possible. Here are the results.

2. The speed of the algorithm PAWZ is statistically 50% higher than that of the algorithm AWZ. For small matrices (100–300) the speed of the algorithm PAWZ is even about 100% greater than the speed of the algorithm AWZ. The speed decreases with the growth of the size for both the algorithms to a certain level. The speed of the algorithm PAWZ stabilizes about 85 Mflops. We present the results in the Figure 2.

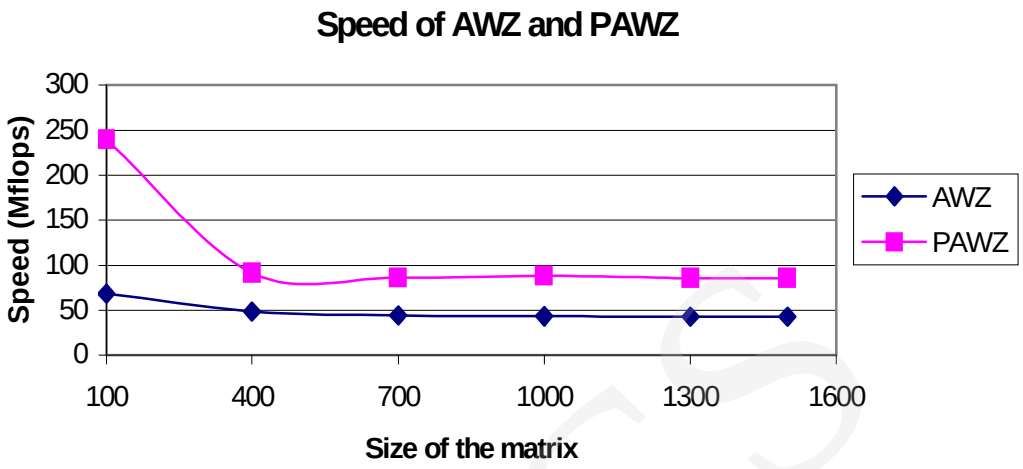

Fig. 2. Speed of AWZ and PAWZ

3. The speed of the algorithm PAWZ is statistically 13% higher than the speed of the algorithm PVAWZ. For big matrices the algorithm PAWZ is about 20% faster than PVAWZ. The results are presented in the Figure 3.

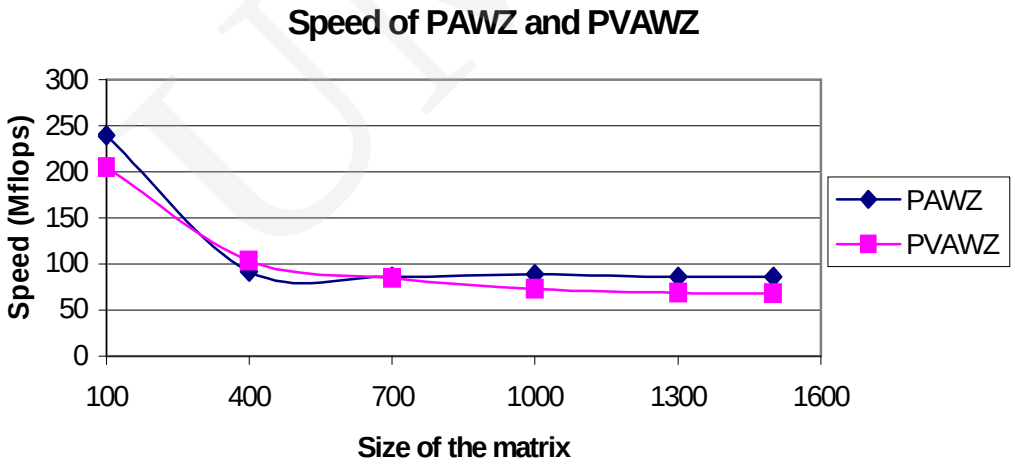

Fig. 3. Speed of PAWZ and PVAWZ

4. The speed of the algorithm PVAWZ is the same as the speed of the algorithm PAWZ. That means that parallelization of VAWZ was without meaning for the speed. The speed stabilizes about 66 Mflops. The results are presented in the Figure 4.

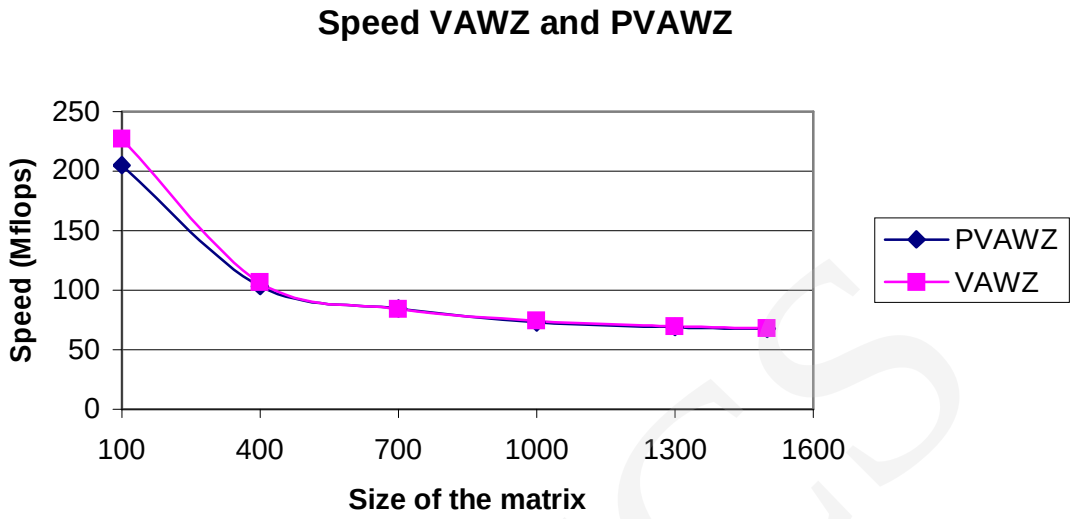

Fig. 4. Speed VAWZ and PVAWZ

From our tests the following inequality (for big matrices) follows:

$$sp(\text{AWZ}) < sp(\text{VAWZ}) \leq sp(\text{PVAWZ}) < sp(\text{PAWZ})$$

where $sp(\text{A})$ denotes the speed of the algorithm A.

And here are the results of our tests for a matrix of the size 2000

	speed of the algorithm (Mflops)			
size of the matrix	AWZ	PAWZ	VAWZ	PVAWZ
2000	38,9282	86,02145	65,28	65,2212

6. Conclusion

We described an efficient algorithm for solving linear systems (1) by WZ factorization. We vectorized (with BLAS1 subroutines) this algorithm. We showed that VAWZ is better than AWZ. Some elements of AWZ and VAWZ were parallelized (using OpenMP standard) and we got algorithms PAWZ and PVAWZ. PAWZ was the most efficient algorithm of the four presented.

References

[1] Chandra Sekhara Rao S., *Existence and uniqueness of WZ factorization*, Parallel Computing, 23 (1997) 1129.

[2] Evans D.J., Barulli M., *BSP linear solver for dense matrices*, Parallel Computing, 24 (1998) 777.

[3] Shanehchi J., The determination of sparse eigensystems and parallel linear system solvers, Ph. D. Thesis, Loughborough University of Technology, (1980).

[4] Anthoine J.L., Chatonnay P., Laiymani D., Nicod J.M., Philippe L., *Parrallel Numerical Computing Using CORBA.*

[5] Evans D.J., Hatzopoulos M., *The parallel solution of linear system*, Int. J. Comp. Math., 7 (1979) 227.

[6] Yalamov P., Evans D.J., *The WZ matrix factorization method*, Parallel Computing, 21 (1995) 1111.
[7] Drozdowski P., *Wprowadzenie do Matlaba*, Politechnika Krakowska im. Tadeusza Kościuszki, Kraków, (1996), in Polish.
[8] *Express C. User's Guide Version 3.0*, ParaSoft Corp., Pasadena Calif., (1990).
[9] *Intel(R) C++ Compiler for Linux Release notes*, version 6.01.
[10] Donngarra J.J., Whaley R.C.: *LAPACK Working Note 94: A User's Guide to the BLACS v 1.1*; http://www.nwtlib.org/lapack/lawns.
[11] *OpenMP C and C++ Application Program Interface*, Version 2.0 March (2002).