Mixed precision iterative refinement techniques for the WZ factorization

Beata Bylina Jarosław Bylina
Institute of Mathematics
Marie Curie-Sklodowska University
Pl. M. Curie-Skłodowskiej 5, 20-031 Lublin, Poland
beatas@hektor.umcs.lublin.pl
jmbylina@hektor.umcs.lublin.pl

***Abstract*—The aim of the paper is to analyze the potential of the mixed precision iterative refinement technique for the WZ factorization. We design and implement a mixed precision iterative refinement algorithm for the WZ factorization with the use of the single (a.k.a. float), double and long double precision. For random dense square matrices with the dominant diagonal we report the performance and the speedup of the solvers using different machines and we investigate the accuracy of obtained solutions. Additionally, the results (performance, speedup and accuracy) for our mixed precision implementation based on the WZ factorization were compared to the similar ones based on the LU factorization.**

I. Introduction

SOLUTION of linear systems of equations of the form:

$$\mathbf{Ax} = \mathbf{b}, \qquad \text{where} \quad \mathbf{A} \in \mathbb{R}^{n\times n}, \quad \mathbf{b} \in \mathbb{R}^n, \qquad (1)$$

is an important and common problem in engineering and scientific computations. One of the direct methods of solving a dense linear system (1) is to factorize the matrix $\mathbf{A}$ into some simpler matrices — and then solving simpler linear systems. The most known factorization is the LU factorization. In this work we study another form of the factorization, namely the WZ factorization. In [5], [6], [7] we showed that there are matrices for which applying the incomplete WZ preconditioning gives better results than the incomplete LU factorization and we also showed the use of the WZ factorization for Markovian models.

One of quite known techniques to accelerate computations is the mixed precision iterative refinement. The iterative refinement is a well-known concept and it was analyzed by [11], [13], [12]. The mixed precision iterative refinement technique is used for the high performance computing [2], for example, for the solution of dense linear systems [3] — the LU factorization was considered among others, for accelerated block-asynchronous iteration methods [1].

The idea of such a refinement is that we perform the most time-consuming computations with the use of the low precision and then we improve the accuracy of the solution with the use of the high precision — by the iterative refinement. The mixed precision method uses properties of modern computer architectures where the single precision computations are about twice faster than the double precision ones — and the same can be observed for the memory access for both precisions.

Here we will modify the WZ solver to use the mixed precision approach. The aim of the paper is to analyze the potential of the mixed precision iterative refinement technique for the WZ factorization. We design and implement a mixed precision iterative refinement algorithm for the WZ factorization and compare its performance, speedup and accuracy with the pure implementation of the WZ factorization with the use of the float, double and long double precisions. We also compare it to an analogous LU solvers (pure ones and with the mixed precision).

The content of the paper is following. In Section II we describe the idea of the WZ factorization [8], [14] and the way the matrix $\mathbf{A}$ is factorized to a product of matrices $\mathbf{W}$ and $\mathbf{Z}$ — such a factorization exists for every nonsingular matrix (with pivoting) what was shown in [8].

Section III provides some mathematical background by outlining the idea of the iterative refinement algorithm.

In Section IV we describe the mixed precision iterative refinement technique for the WZ factorization. We present the algorithm for matrices which can be factorized without pivoting, for example, strictly diagonally dominant ones (as it was proved in [8]) and we give details about the implementation of the mixed precision iterative refinement for the WZ factorization.

In Section V we present the results of our experiments. We analyze the performance of our algorithm and its speedup. We study the influence of the size of the matrix on the achieved numerical accuracy.

Section VI is a summary of our experiments.

II. WZ Factorization

Here we describe shortly the WZ factorization usage to solve (1). The WZ factorization is described in [8], [10]. Assume that the $\mathbf{A}$ is a square nonsingular matrix. We are to find matrices $\mathbf{W}$ and $\mathbf{Z}$ that fulfill $\mathbf{WZ} = \mathbf{A}$ and the matrices $\mathbf{W}$ and $\mathbf{Z}$ consist of the following columns $\mathbf{w}_i$ and rows $\mathbf{z}_i^T$

This work was partially supported within the project N N516 479640 of the Ministry of Science and Higher Education of the Polish Republic (MNiSW) “Modele dynamiki transmisji, sterowania zatłoczeniem i jakością usług w Internecie”.

respectively:

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

where

$$\begin{aligned}
m &= \lfloor (n-1)/2 \rfloor, \\
p &= \lfloor (n+1)/2 \rfloor, \\
q &= \lceil (n+1)/2 \rceil.
\end{aligned}$$

(see also Figure 1).

After the factorization we can solve two linear systems:

$$\begin{aligned}
\mathbf{Wy} &= \mathbf{b}, \\
\mathbf{Zx} &= \mathbf{y}
\end{aligned}$$

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

III. Refinement

Let $\mathbf{x}_{cr}$ be the exact solution of the system (1):

$$\mathbf{A}\mathbf{x}_{cr} = \mathbf{b} \tag{2}$$

and $\mathbf{x}_{cm}$ be the machine-computed solution, thus with some rounding error — which we denote by $\mathbf{e}$ (all $\mathbf{x}_{cr}$, $\mathbf{x}_{cm}$, $\mathbf{e}$ are vectors of the size n).

Then, we can write:

$$\mathbf{x}_{cm} = \mathbf{x}_{cr} - \mathbf{e} \tag{3}$$

Let

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}_{cm} \tag{4}$$

be a residual vector for the not exact solution $\mathbf{x}_{cm}$. Using (3) for $\mathbf{x}_{cm}$ in (4) we get:

$$\mathbf{r} = \mathbf{b} - \mathbf{A}(\mathbf{x}_{cr} - \mathbf{e})$$

and then (from (2)):

$$\mathbf{r} = \mathbf{A}\mathbf{e}. \tag{5}$$

Now, we can compute the error vector $\mathbf{e}$ from a linear system (5) and find a new, better solution of (1):

$$\mathbf{x}'_{cm} = \mathbf{x}_{cm} + \mathbf{e}.$$

However, the vector $\mathbf{e}$ as a solution of (5) is also prone to rounding errors, so the new $\mathbf{x}'_{cm}$ is also not exact — although better — and it can be further improved iteratively with the same process. This routine is known as the iterative refinement.

Algorithm 1 describes steps of such an iterative refinement for the solution of the linear system (1), with the use of the WZ factorization. The computational complexity is also given for every step. The stop condition is given by the infinity norm of the residual vector:

$$||\mathbf{r}||_\infty = \max_{1 \le i \le n} |r_i|,$$

and the refinement stops when there is no further improvement (that is why we return $\mathbf{x}$, not $\mathbf{x}'$).

Algorithm 1 The iterative refinement technique for the WZ factorization

```
Require: $\mathbf{A}$, $\mathbf{b}$
Ensure: $\mathbf{x} \leftarrow \mathbf{A}^{-1}\mathbf{b}$
 1: $\mathbf{WZ} \leftarrow \mathbf{A}$                                   $\{O(n^3)\}$
 2: Solve the equation $\mathbf{Wy} = \mathbf{b}$                   $\{O(n^2)\}$
 3: Solve the equation $\mathbf{Zx} = \mathbf{y}$                   $\{O(n^2)\}$
 4: $\mathbf{r} \leftarrow \mathbf{Ax} - \mathbf{b}$                $\{O(n^2)\}$
 5: $\varepsilon \leftarrow ||\mathbf{r}||_\infty$                  $\{O(n)\}$
 6: loop
 7:     Solve the equation $\mathbf{Wy} = \mathbf{r}$               $\{O(n^2)\}$
 8:     Solve the equation $\mathbf{Zp} = \mathbf{y}$               $\{O(n^2)\}$
 9:     $\mathbf{x}' \leftarrow \mathbf{x} + \mathbf{p}$            $\{O(n)\}$
10:     $\mathbf{r} \leftarrow \mathbf{Ax}' - \mathbf{b}$           $\{O(n^2)\}$
11:     $\varepsilon' \leftarrow ||\mathbf{r}||_\infty$             $\{O(n)\}$
12:     if $\varepsilon' \ge \varepsilon$ : return $\mathbf{x}$      $\{O(1)\}$
13:     $\mathbf{x} \longleftrightarrow \mathbf{x}'$        $\{O(1)$, only references swapped$\}$
14:     $\varepsilon \leftarrow \varepsilon'$                        $\{O(1)\}$
15: end loop
```

IV. Mixed Precision

The next algorithm, Algorithm 2, describes the use of the mixed precision technique with the iterative refinement of the solution. Every step is also labeled with its precision.

The only operations performed with the use of the double (or long double) precision are:

- matrix-vector operations with the complexity $O(n^2)$ — Steps 7 and 15 (computing the residual vector);
- vector operations with the complexity $O(n)$ — Steps 8, 16 (computing the norm) and 14 (applying the correction);
- scalar operations with the complexity $O(1)$ — Steps 17 and 19;
- conversions — almost all with the complexity $O(n)$ — the only exception ($O(n^2)$) is Step 1, but it is done only once, before the loop.

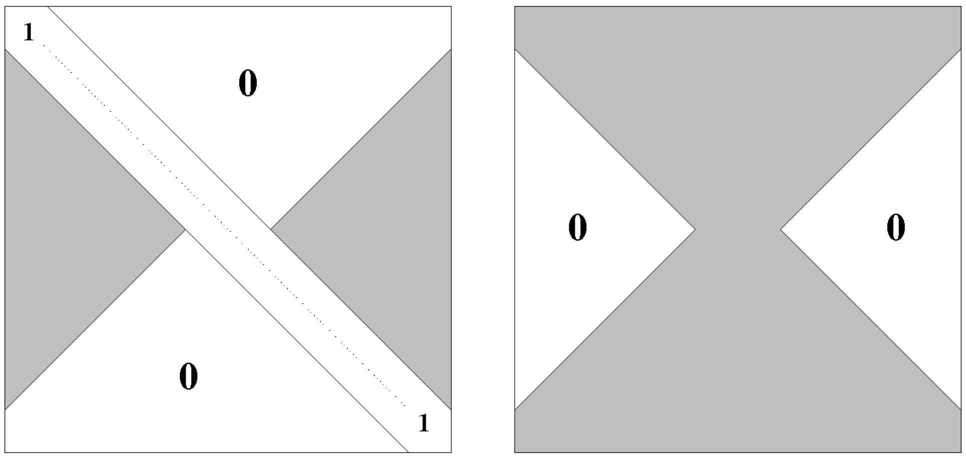

Fig. 1. Structures of the matrices $\mathbf{W}$ (left) and $\mathbf{Z}$ (right)

Algorithm 2 The mixed precision iterative refinement technique for the WZ factorization

```
Require: $\mathbf{A}_d$, $\mathbf{b}_d$
Ensure: $\mathbf{x}_d \leftarrow \mathbf{A}_d{}^{-1}\mathbf{b}_d$
 1: $\mathbf{A}_s \leftarrow \mathbf{A}_d$                                  {conversion}
 2: $\mathbf{b}_s \leftarrow \mathbf{b}_d$                                  {conversion}
 3: $\mathbf{W}_s\mathbf{Z}_s \leftarrow \mathbf{A}_s$                      {$O(n^3)$, single}
 4: Solve the equation $\mathbf{W}_s\mathbf{y}_s = \mathbf{b}_s$           {$O(n^2)$, single}
 5: Solve the equation $\mathbf{Z}_s\mathbf{x}_s = \mathbf{y}_s$           {$O(n^2)$, single}
 6: $\mathbf{x}_d \leftarrow \mathbf{x}_s$                                  {conversion}
 7: $\mathbf{r}_d \leftarrow \mathbf{A}_d\mathbf{x}_d - \mathbf{b}_d$       {$O(n^2)$, (long) double}
 8: $\varepsilon \leftarrow ||\mathbf{r}_d||_\infty$                        {$O(n)$, (long) double}
 9: loop
10:     $\mathbf{r}_s \leftarrow \mathbf{r}_d$                              {conversion}
11:     Solve the equation $\mathbf{W}_s\mathbf{y}_s = \mathbf{r}_s$       {$O(n^2)$, single}
12:     Solve the equation $\mathbf{Z}_s\mathbf{p}_s = \mathbf{y}_s$       {$O(n^2)$, single}
13:     $\mathbf{p}_d \leftarrow \mathbf{p}_s$                              {conversion}
14:     $\mathbf{x}'_d \leftarrow \mathbf{x}_d + \mathbf{p}_d$              {$O(n)$, (long) double}
15:     $\mathbf{r}_d \leftarrow \mathbf{A}_d\mathbf{x}'_d - \mathbf{b}_d$  {$O(n^2)$, (long) double}
16:     $\varepsilon' \leftarrow ||\mathbf{r}_d||_\infty$                   {$O(n)$, (long) double}
17:     if $\varepsilon' \geq \varepsilon$ : return $\mathbf{x}_d$          {$O(1)$, (long) double}
18:     $\mathbf{x}_d \longleftrightarrow \mathbf{x}'_d$                    {$O(1)$, only references swapped}
19:     $\varepsilon \leftarrow \varepsilon'$                               {$O(1)$, (long) double}
20: end loop
```

All the other computations are single precision ones.

We denote the matrices and vectors stored in the (long) double precision with the $_d$ subscript and the matrices and vectors stored in the single precision with the $_s$ subscript. So, the input data are the matrix $\mathbf{A}_d$ and the vector $\mathbf{b}_d$ ((long) double precision). The output vector $\mathbf{x}_d$ is also stored in the (long) double precision.

The coefficient matrix $\mathbf{A}_d$ is converted to the single precision for the WZ factorization and denoted as $\mathbf{A}_s$. Some vectors in the algorithm are also converted between single and (long) double precision (that is why we have: $\mathbf{b}_d$ and $\mathbf{b}_s$; $\mathbf{x}_d$ and $\mathbf{x}_s$; $\mathbf{r}_d$ and $\mathbf{r}_s$; $\mathbf{p}_d$ and $\mathbf{p}_s$).

The method used in Algorithm 2 can give a significant improvement for the solution of a linear system, because the cost of every iteration is very small comparing to the cost of the factorization.

The disadvantage of this approach is a lot larger memory requirement — a great deal of data are to be duplicated in both precisions. It consumes up to 50% more memory than it is used in usual double precision solution.

V. Numerical Experiments

In the experiment, we analyze how the use of the mixed precision iterative refinement techniques for the WZ factorization influences the performance, the speedup and the accuracy of the WZ solver for the linear equation system. In this section we test the performance, the speedup and the accuracy on three devices of different architectures. Additionally, we compare properties of the WZ solver with the LU solver. For both kinds of factorization we consider five implementations:

- a traditional single precision implementation;
- a traditional double precision implementation;
- a traditional long double precision implementation;
- a mixed precision implementation with double precision refinement (denoted as mix(double));
- a mixed precision implementation with long double precision refinement (denoted as mix(long double)).

The former three are made according to Algorithm 1 and the latter two are made according to Algorithm 2. All the implementations were sequential (single-threaded).

All the computations were carried out for dense random matrices with a dominant diagonal. The sizes of the matrices were from 500 up to 9000.

A. Environment

The architectures used for tests are shown in Table I. All the machines worked under the Debian GNU/Linux 6.0 operating system and the programs were compiled with the use of the GCC compiler (ver. 4.7.2, compiler command: `g++ -O3`).

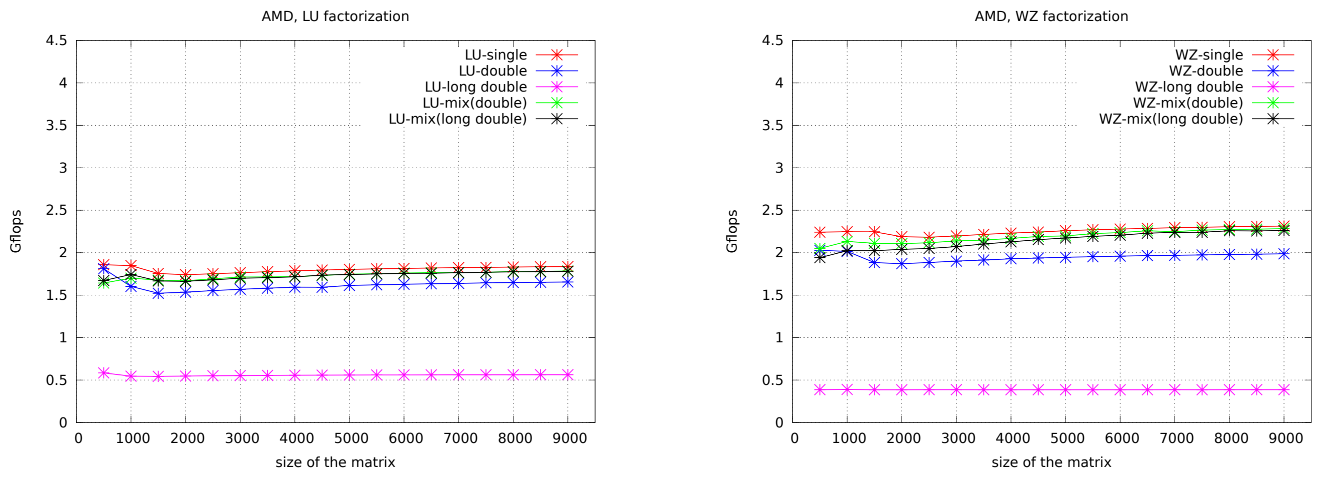

Fig. 2. The performance of the LU (left) and WZ (right) solver on the AMD architecture

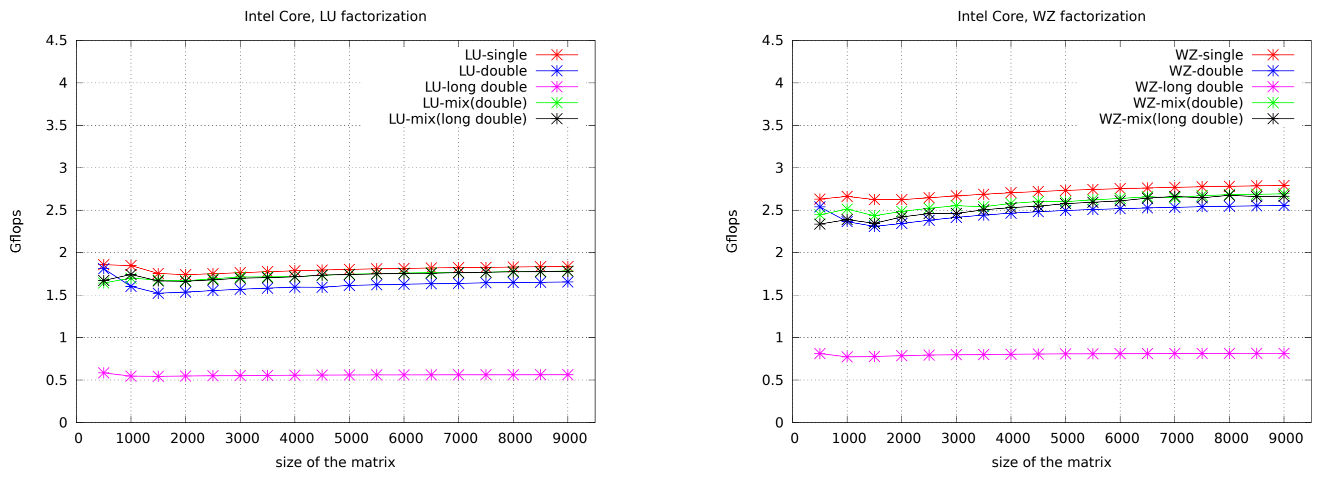

Fig. 3. The performance of the LU (left) and WZ (right) solver on the Intel Core architecture

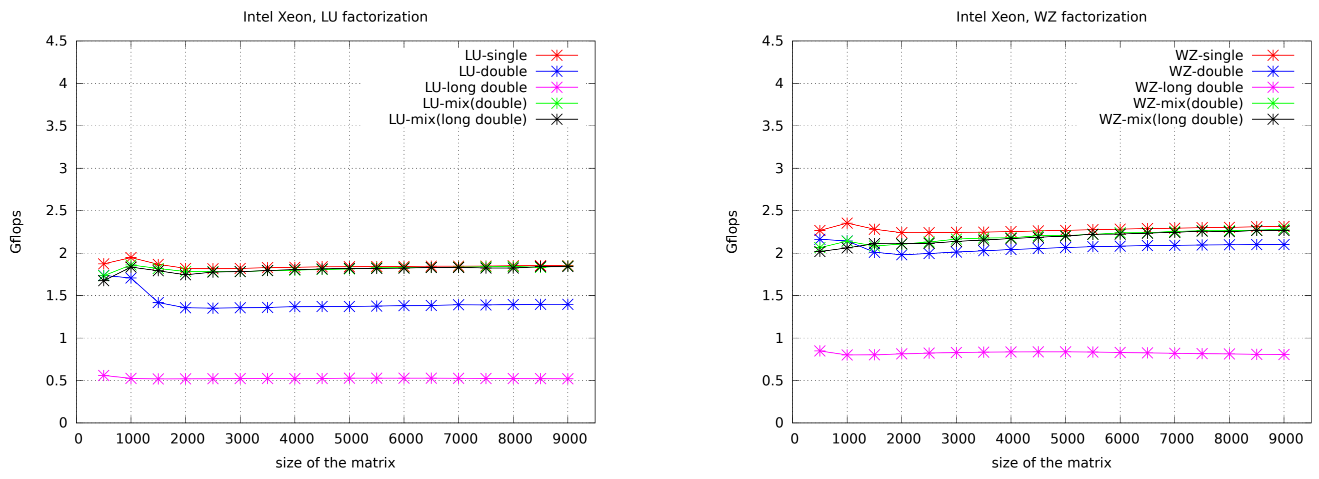

Fig. 4. The performance of the LU (left) and WZ (right) solver on the Intel Xeon architecture

B. *Performance*

Figures 2, 3, 4 show the performance of the single-core implementations of the LU and WZ solvers on the given architectures. The performance is based on the number of floating-point operations in the LU solver $\left(\frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{7}{6}n\right)$.

We see that:

- the architecture has almost no impact on the performance — however, not mixed implementations on AMD are somewhat slower;
- the size of the matrix has no impact on the performance, either;
- the WZ solver performs better than the LU solver;
- the long double implementation is always the slowest, the others perform quite similarly — even (what is very important) the mix(long double) implementation.

C. *Speedup*

Figures 5, 6, 7 show the speedup of the same implementations. We labeled our speedups by S(F)-P where:

- $\mathsf{F} \in \{\mathsf{LU}, \mathsf{WZ}\}$ is the kind of the factorization used;
- $\mathsf{P} \in \{\mathsf{double}, \mathsf{long\ double}\}$ is the precision of the result.

Thus, S(F)-P denotes the speedup achieved by the mix(P) implementation over the P implementation — while both are conducted with the use of the F factorization.

We see that:

- the architecture has some impact on the speedup: the best is for the AMD architecture — because that architecture gives somewhat slower performance for double and long double implementations;
- for very small problem sizes, the cost of even a few iterative refinement iterations is high compared to the cost of the factorization and thus, the mix(double) implementations are less efficient than the double ones;
- the LU solvers have higher speedups than the WZ solvers for all architectures — because the original LU solver performs slightly worse than the WZ one;
- if the problem size is big enough, the mix(double) implementation can provide a speedup of up to 1.3 and the mix(long double) — even up to 7.

D. *Accuracy*

Figures 8, 9, 10 show the accuracy of the implementations. We define the measure of the accuracy as

$$\mathsf{accu} = -\log_{10} ||\mathbf{Ax} - \mathbf{b}||_\infty .$$

We see that:

TABLE I
HARDWARE PROPERTIES OF THE TEST MACHINES

AMD	CPU Host memory	AMD FX-8120 3.1 GHz 16 GB
Intel Core	CPU Host memory	Intel Core I7 2670QM 2.2 GHz 8 GB
Intel XEON	CPU Host memory	Intel Xeon X5650 2.67GHz 48 GB

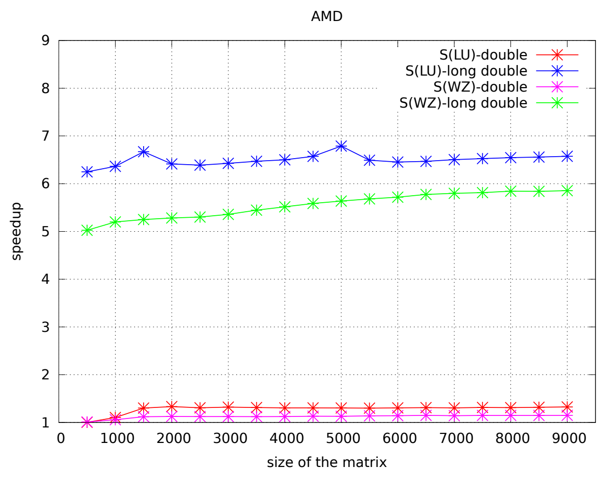

Fig. 5. The speedup of the LU and WZ solvers on the AMD

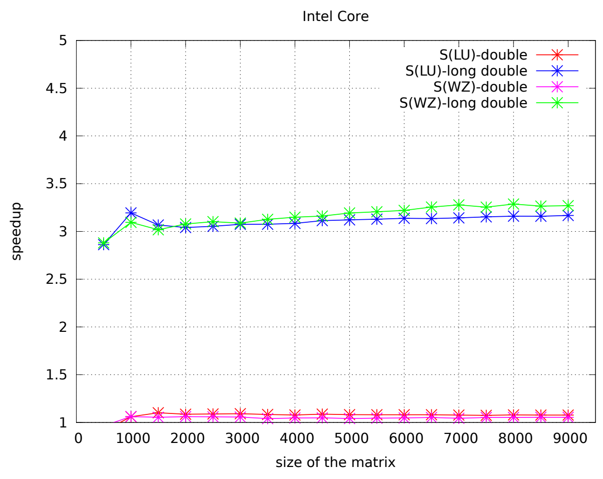

Fig. 6. The speedup of the LU and WZ solvers on the Intel Core

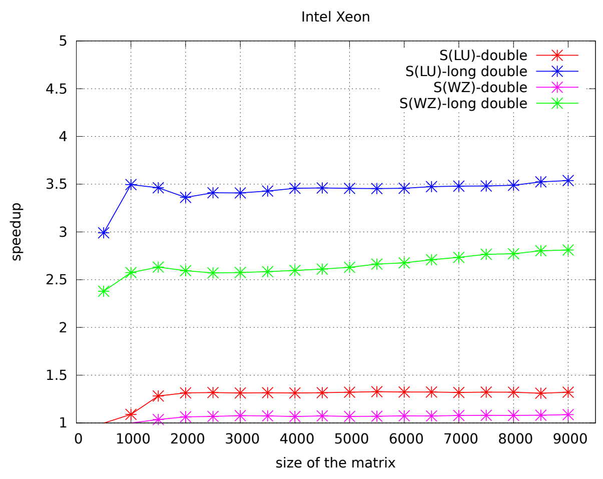

Fig. 7. The speedup of the LU and WZ solvers on the Intel Xeon

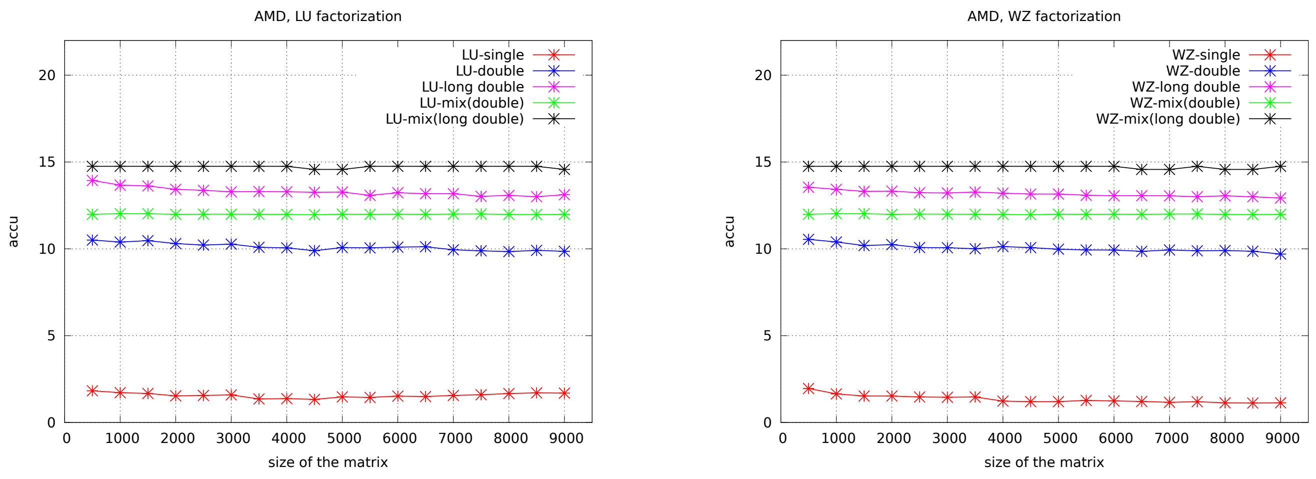

Fig. 8. The accuracy of the LU (left) and WZ (right) solver on the AMD architecture

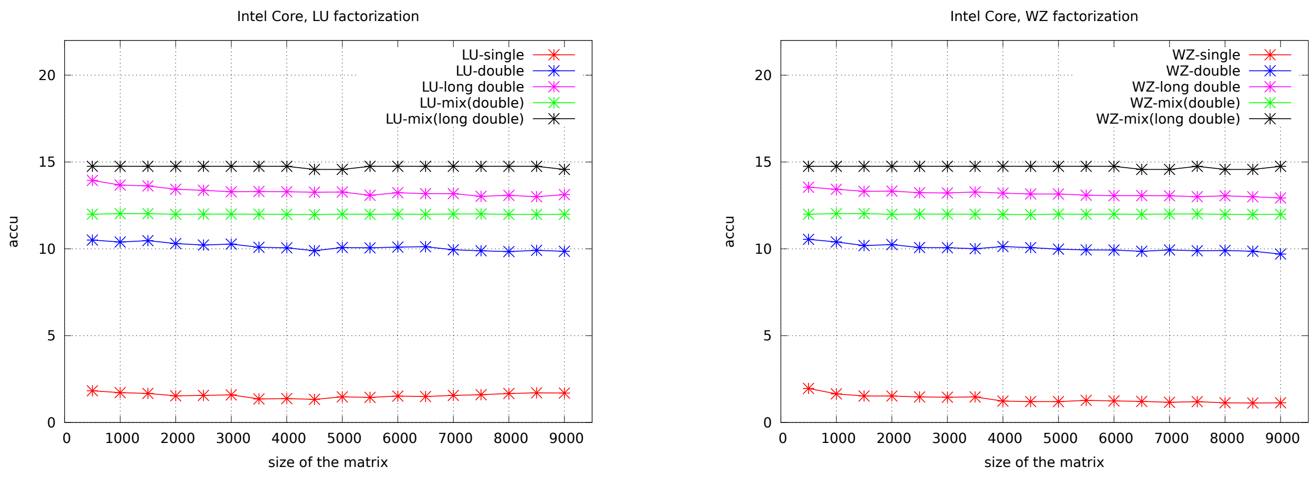

Fig. 9. The accuracy of the LU (left) and WZ (right) solver on the Intel Core architecture

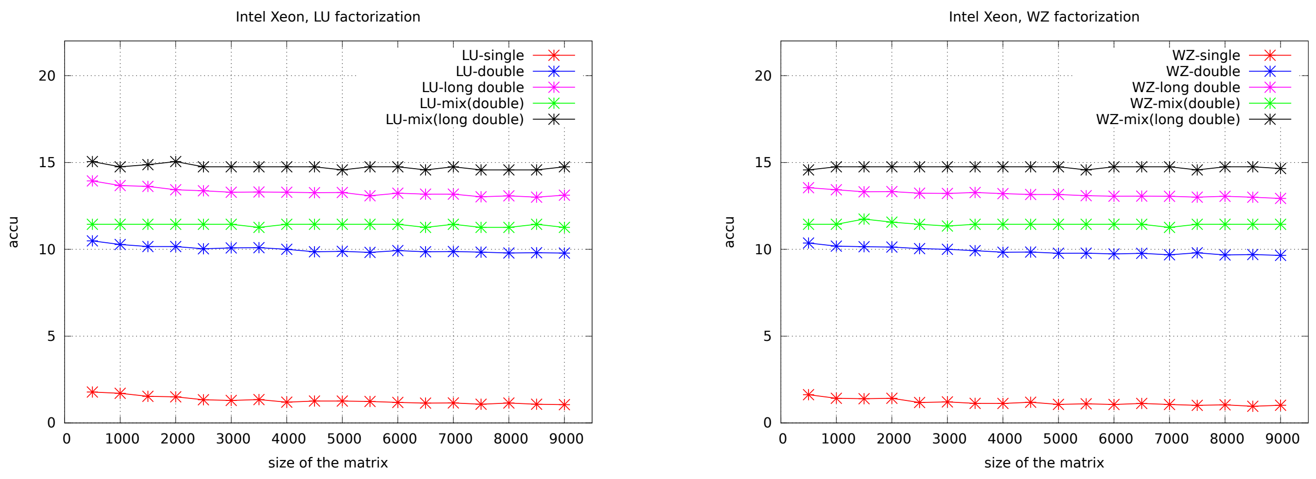

Fig. 10. The accuracy of the LU (left) and WZ (right) solver on the Intel Xeon architecture

- the architecture has no impact on the accuracy;
- the size of the matrix and the type of the factorization has almost no impact on the accuracy, either;
- the worst accuracy we get is (of course) for the single implementation; the best (about 10^{-15}) — for the long double one;
- the mixed precision significantly improves the accuracy.

The number of iterations needed for our mixed precision method to outdo the accuracy of the (long) double precision solver is not too high and is about 5–6 iterations, somewhat less on the AMD architecture (about 3–4 iterations).

VI. Conclusion

In this article we described an iterative refinement algorithm for the WZ solver with the use of the mixed precision technique and investigated properties of this new algorithm. We compared the mixed precision iterative refinement for the WZ factorization with a similar algorithm for the LU factorization.

Both the types of algorithms gave similar accuracy. However, the performance was better for the WZ solvers but the higher speedup was achieved for the LU solver.

These experiments show that the mixed precision iterative refinement method can run faster than the (long) double precision solver — delivering the same (or even better) accuracy as the (long) double precision one for both the factorizations. Moreover, the experiments also show that the mixed precision iterative refinement method for the long double precision solver is much faster than the traditional one (up to 7 times) — with the same or better accuracy.

The results do not depend significantly on the size of the matrix.

The approach presented here causes a significant acceleration of solving the linear systems with the use of direct methods and we think that the similar problems on different architectures (as GPU, for example) could be also improved.

References

[1] H. Anzt, P. Luszczek, J. Dongarra, V. Heuveline: GPU-Accelerated Asynchronous Error Correction for Mixed Precision Iterative Refinement, *Euro-Par 2012*, pp. 908–919.

[2] M. Baboulin, A. Buttari, J. J. Dongarra, J. Langou, J. Langou, P. Luszczek, J. Kurzak, S. Tomov: Accelerating scientific computations with mixed precision algorithms, *Computer Physics Communications* 180(12) (2009), pp. 2526–2533.

[3] A. Buttari, J. J. Dongarra, J. Langou, J. Langou, P. Luszczek, J. Kurza: Mixed precision iterative refinement techniques for the solution of dense linear systems. *Int. J. of High Performance Computing and Applications* 21(4) 2007, pp. 457–466.

[4] B. Bylina, J. Bylina: Analysis and Comparison of Reordering for Two Factorization Methods (LU and WZ) for Sparse Matrices, *Lecture Notes in Computer Science* **5101**, Springer-Verlag Berlin Heidelberg 2008, pp. 983–992.

[5] B. Bylina, J. Bylina: Incomplete WZ Factorization as an Alternative Method of Preconditioning for Solving Markov Chains, *Lecture Notes in Computer Science* **4967**, Springer-Verlag Berlin Heidelberg 2008, 99–107.

[6] B. Bylina, J. Bylina: Influence of preconditioning and blocking on accuracy in solving Markovian models, *International Journal of Applied Mathematics and Computer Science* 19 (2) (2009), pp. 207–217.

[7] B. Bylina, J. Bylina: The Vectorized and Parallelized Solving of Markovian Models for Optical Networks, *Lecture Notes in Computer Science* **3037**, Springer-Verlag Berlin Heidelberg 2004, 578–581.

[8] S. Chandra Sekhara Rao: Existence and uniqueness of WZ factorization, *Parallel Computing* **23** (1997), pp. 1129–1139.

[9] D. J. Evans, M. Barulli: BSP linear solver for dense matrices, *Parallel Computing* **24** (1998), pp. 777–795.

[10] D. J. Evans, M. Hatzopoulos: The parallel solution of linear system, *Int. J. Comp. Math.* **7** (1979), pp. 227–238.

[11] C. B. Moler: Iterative refinement in floating point. *J. ACM* **14(2)** (1967), pp. 316–21.

[12] G. W. Stewart: Introduction to Matrix Computations, Academic Press, 1973.

[13] J. H. Wilkinson: Rounding Errors in Algebraic Processes, Prentice-Hall, 1963.

[14] P. Yalamov, D. J. Evans: The WZ matrix factorization method, *Parallel Computing* **21** (1995), pp. 1111–1120.

[15] `http://software.intel.com/en-us/articles/intel-mkl/`