Forward Stability of Iterative Refinement with a Relaxation for Linear Systems

Alicja Smoktunowicz*, Jakub Kierzkowski and Iwona Wróbel

Faculty of Mathematics and Information Science, Warsaw University of Technology, Koszykowa 75, 00-662 Warsaw, Poland

Abstract: Stability analysis of Wilkinson's iterative refinement method IR(ω) with a relaxation parameter ω for solving linear systems is given. It extends existing results for $\omega = 1$, i.e., for Wilkinson's iterative refinement method. We assume that all computations are performed in fixed (working) precision arithmetic. Numerical tests were done in *MATLAB* to illustrate our theoretical results. A particular emphasis is given on convergence of iterative refinement method with a relaxation. A preliminary error analysis of the Algorithm IR(ω) was given in [11]. Our opinion is opposite to that given in [11], since our experiments show that the choice $\omega = 1$ is the best choice from the point of numerical stability.

Keywords: Iterative refinement, numerical stability, condition number.

1. INTRODUCTION

We consider the system $Ax = b$, where $A \in R^{n\times n}$ is nonsingular and $b \in R^n$. Iterative refinement techniques for linear systems of equations are very useful in practice and the literature on this subject is very rich, see [1], [4]– [11].

The idea of relaxing the iterative refinement step is the following. We require a basic linear equation solver S for $Ax = b$ which uses a factorization of A into simple factors (e.g., triangular, block-triangular etc.). Such factorization is used again in the next steps of iterative refinement. Wilkinson's iterative refinement method with a relaxation IR(ω) consists of three steps.

Algorithm IR(ω)

Given $\omega > 0$. Let x_0 be computed by the solver S.

For $k = 0,1,2,\ldots$, the kth iteration consists of the three steps:

1. Compute $r_k = b - Ax_k$.

2. Solve $Ap_k = r_k$ for p_k by the basic solution solver S.

3. Add the correction, $x_{k+1} = x_k + \omega p_k$.

Clearly, $\omega = 1$ corresponds to Wilkinson's iterative refinement method [10]. Wu and Wang [11] proposed this method for $\omega = \frac{h}{h+1}$, where $h > 0$ (i.e., for $0 < \omega < 1$). They developed the method as the numerical integration of a dynamic system with step size h. A preliminary error analysis of the Algorithm IR(ω) was given in [11] for $0 < \omega < 1$, assuming that the extended precision is used for computing the residual vectors r_k. Wu and Wang considered only Gaussian elimination as a solver S.

The purpose of this paper is to analyze the convergence of this method for $0 < \omega < 2$ and to show with examples that the choice $\omega = 1$ is the best choice from the point of numerical stability.

Notice that for arbitrary $\omega > 0$, the IR(ω) method is a stationary method (in the theory) and we have $p_k = A^{-1}r_k = x^* - x_k$, so $x_{k+1} - x^* = (1-\omega)(x_k - x^*)$, $k = 0,1,\ldots$, where x^* is the exact solution to $Ax = b$. We see that the sequence $\{x_k\}$ is convergent for arbitrary initial x_0 if and only if $0 < \omega < 2$. For $\omega = 1$ (Wilkinson's iterative refinement) x_1 will be the exact solution x^*. It is interesting to check the influence on the relaxation parameter ω on numerical properties of the algorithm IR(ω), assuming that all computations are performed only in the working (fixed) precision.

Throughout the paper we use only the 2-norm and assume that all computations are performed in the working (fixed) precision. We use a floating point arithmetic which satisfies the IEEE floating point standard. For two floating point numbers a and b we have

$$f\ell(a\lozenge b) = (a\lozenge b)(1+\Delta), \quad |\Delta| \le \varepsilon_M$$

for results in the normalized range, where $\lozenge$ denotes any of the elementary scalar operations $+,-,*,/$ and ε_M is machine precision.

*Address correspondence to this author at the Faculty of Mathematics and Information Science, Warsaw University of Technology, Koszykowa 75, 00-662 Warsaw, Poland; Tel: +48222347988; Fax: +48226257460;
E-mail: smok@mini.pw.edu.pl

In this paper we present a comparison of Wilkinson's iterative refinement method with a relaxation IR(ω) from the point of view of numerical stability. More precisely, we say that the computed $\tilde{x}$ in floating point arithmetic is a **forward stable** solution to $Ax = b$ if

$$\| \tilde{x} - x^* \| \le O(\varepsilon_M)\kappa(A) \| x^* \| . \quad (1)$$

Throughout this paper, $\| \cdot \|$ is the matrix or vector two–norm depending upon context, and $\kappa(A) = \| A^{-1} \| \| A \|$ denotes the standard condition number of the matrix A.

A stronger property than forward stability is backward stability. It means that the computed $\widetilde{x}$ in floating point arithmetic is the exact solution of a slightly perturbed system

$$(A + \Delta A)\tilde{x} = b, \quad \| \Delta A \| \le O(\varepsilon_M) \| A \| . \quad (2)$$

Our analysis is similar in spirit to [4]-[6]. Jankowski and Woz'niakowski [6] prove that an arbitrary solver S which satisfies (3), supported by iterative refinement, is normwise forward stable as long as A is not too ill-conditioned (say, $\varepsilon_M \kappa(A) < 1$), and is normwise backward stable under additional condition $q\kappa(A) < 1$. We extend their results for the algorithm IR(ω), see Theorems 2.1.

The paper is organized as follows. A proof of forward stability of IR(ω) is given in Section 2. In Section 3, we present some numerical experiments that illustrate our theoretical results.

2. FORWARD STABILITY OF IR(ω)

We require a basic linear equation solver S for $Ax = b$ such that the computed solution $\tilde{x}$ by S satisfies

$$\| \tilde{x} - x^* \| \le q \| x^* \|, \quad q \le 0.1. \quad (3)$$

We make a standard assumption that the matrix-vector multiplication is backward stable. Then the computed residual vector $\tilde{r} = f\ell(b - A\tilde{x})$ satisfies

$$\tilde{r} = b - A\tilde{x} + \Delta r, \quad \| \Delta r \| \le L(n)\varepsilon_M (\| b \| + \| A \| \| \tilde{x} \|), \quad (4)$$

where $L(n)$ is a modestly growing function on n.

We start with the following lemma.

Lemma 2.1 Let IR(ω) for $\omega \in (0,2)$ be applied to the nonsingular linear system $Ax = b$ using the solver S satisfying (3)-(4). Let $\tilde{x}_k$, $\tilde{r}_k$ and $\tilde{p}_k$ denote the computed vectors in floating point arithmetic. Assume that

$$\varepsilon_M \le 0.01, \quad L(n)\varepsilon_M \kappa(A) \le 0.01 \quad (5)$$

and

$$| 1 - \omega | + \omega q \le 0.6. \quad (6)$$

Then for $k = 0,1,\ldots$ we have

$$\| \tilde{x}_k - x^* \| \le q_k \| x^* \|, \quad q_k \le 0.1, \quad (7)$$

where

$$q_{k+1} = (| 1 - \omega | + q\omega)q_k + 2.31\omega L(n)\varepsilon_M \kappa(A) + 1.64\varepsilon_M , \quad (8)$$

with $q_0 = q$.

Proof. Assume that (7) holds for k. We prove that it holds also for $k+1$, i.e. $\| \tilde{x}_{k+1} - x^* \| \le q_{k+1} \| x^* \|$, where $q_{k+1} \le 0.1$ and q_{k+1} satisfies (8).

Under assumption (4), the computed vectors $\widetilde{r}_k$ satisfy

$$\tilde{r}_k = b - A\tilde{x}_k + \Delta r_k , \quad \| \Delta r_k \| \le \varepsilon_M L(n)(\| b \| + \| A \| \| \tilde{x}_k \|). \quad (9)$$

Under assumption (3) we have

$$\tilde{p}_k = p_k^* + \Delta p_k , \quad p_k^* = A^{-1}\tilde{r}_k , \quad \| \Delta p_k \| \le q \| p_k^* \| . \quad (10)$$

Standard error analysis shows

$$\tilde{x}_{k+1} = (I + D_k^{(1)})(\tilde{x}_k + (I + D_k^{(2)})\omega\tilde{p}_k), \quad \| D_k^{(i)} \| \le \varepsilon_M . \quad (11)$$

By inductive assertion, we have $\| \tilde{x}_k - x^* \| \le q_k \| x^* \|$. Hence

$$\| \tilde{x}_k \| = \| x^* + (\tilde{x}_k - x^*) \| \le \| x^* \| + \| \tilde{x}_k - x^* \| \le (1 + q_k) \| x^* \| .$$

Similarly, from (10) it follows that $\| \tilde{p}_k \| \le (1+q) \| p_k^* \|$, thus

$$\| \tilde{x}_k \| \le 1.1 \| x^* \|, \quad \| \tilde{p}_k \| \le 1.1 \| p_k^* \| . \quad (12)$$

From (9) and the inequality $\| b \| = \| Ax_* \| \le \| A \| \| x_* \|$ it can be seen that

$$\tilde{r}_k = b - A\tilde{x}_k + \Delta r_k , \quad \| \Delta r_k \| \le 2.1L(n)\varepsilon_M \| A \| \| x^* \| . \quad (13)$$

We have

$$p_k^* = A^{-1}\tilde{r}_k = x^* - \tilde{x}_k + \xi_k , \quad \xi_k = A^{-1}\Delta r_k . \quad (14)$$

This together with (13) implies the bounds

$$\| p_k^* \| \le \| \tilde{x}_k - x^* \| + \| \xi_k \|, \quad \| \xi_k \| \le 2.1 L(n) \varepsilon_M \kappa(A) \| x^* \|. \tag{15}$$

Now our task is to bound the error $\| \tilde{x}_{k+1} - x^* \|$. For simplicity, we define $D_k^{(3)}$ such that

$$I + D_k^{(3)} = (I + D_k^{(1)})(I + D_k^{(2)}).$$

Clearly, $\| D_k^{(3)} \| \le 2\varepsilon_M + \varepsilon_M^2$, so from (11) we get

$$\tilde{x}_{k+1} = (\tilde{x}_k + \omega \tilde{p}_k) + \eta_k, \quad \| \eta_k \| \le \varepsilon_M \| \tilde{x}_k \| + (2\varepsilon_M + \varepsilon_M^2)\omega \| \tilde{p}_k \|. \tag{16}$$

This together with (10) and (14) gives the identity

$$\tilde{x}_{k+1} - x^* = (1-\omega)(\tilde{x}_k - x^*) + \eta_k + \omega(\xi_k + \Delta p_k).$$

Taking norms and using (10), we obtain

$$\| \tilde{x}_{k+1} - x^* \| \le |1-\omega| \| \tilde{x}_k - x^* \| + \| \eta_k \| + \omega \| \xi_k \| + \omega q \| p_k^* \|. \tag{17}$$

First we estimate $\| \eta_k \|$. Since $\| \tilde{x}_k - x^* \| \le 0.1 \| x^* \|$, so by assumption (5) we obtain from (15) the bounds

$$\| \xi_k \| \le 0.021 \| x^* \|, \quad \| p_k^* \| \le 0.121 \| x^* \|. \tag{18}$$

From (12) and (16) we have $\| \eta_k \| \le 1.1\varepsilon_M (\| x^* \| + (2+\varepsilon_M)\omega \| p_k^* \|)$. Now we apply (5) and (18). Since $\omega < 2$, we see that $\| \eta_k \| \le 1.64 \varepsilon_M \| x^* \|$. Therefore,

$$\omega \| \xi_k \| + \| \eta_k \| \le \omega 2.1 L(n) \varepsilon_M \kappa(A) \| x^* \| + 1.64 \varepsilon_M \| x^* \|$$

and by (15) we get

$$\omega q \| p_k^* \| \le \omega q \| \tilde{x}_k - x^* \| + \omega q 2.1 L(n) \varepsilon_M \kappa(A) \| x^* \|.$$

Hence, from (17) and by (5)-(6) we finally obtain

$$\| \tilde{x}_{k+1} - x^* \| \le (|1-\omega| + \omega q) \| \tilde{x}_k - x^* \| + 2.31 \omega L(n) \varepsilon_M \kappa(A) + 1.64 \varepsilon_M \| x^* \|.$$

We conclude that, $\| \tilde{x}_{k+1} - x^* \| \le q_{k+1} \| x^* \|$ with q_{k+1} defined in (8). It remains to prove that $q_{k+1} \le 0.1$. By assumptions (5) and (6) and using the fact that $q_k \le 0.1$, we see that $q_{k+1} \le 0.6 * 0.1 + (0.0231 + 0.0164)$, so $q_{k+1} \le 0.1$. This completes the proof.

Theorem 2.1 Under the assumptions of Lemma 2.1 the algorithm IR(ω) is forward stable for $\omega \in (0,2)$. There exists k^* depending only on n such that for every $k \ge k^*$ the following inequality holds

$$\| \tilde{x}_k - x^* \| \le (11.6 L(n) + 4.2) \varepsilon_M \kappa(A) \| x^* \|. \tag{19}$$

Proof. We apply the results of Lemma 2.1. Notice that from (7)-(8) and by assumptions (5) it follows that

$$q_{k+1} \le q_k 0.6 + 2.31 \omega L(n) \varepsilon_M \kappa(A) + 1.64 \varepsilon_M.$$

Since $\omega < 2$ and $1 \le \kappa(A)$, we get

$$q_{k+1} \le q_k 0.6 + (4.62 L(n) + 1.64) \varepsilon_M \kappa(A).$$

From this it follows that

$$q_{k+1} \le (0.6)^k + \frac{4.62 L(n) + 1.64}{1 - 0.6} \varepsilon_M \kappa(A).$$

From this (19) follows immediately.

3. NUMERICAL EXPERIMENTS

In this section we present numerical experiments that show the comparison of the IR(ω) for different values of ω. All tests were performed in *MATLAB* version 8*.4.0.150421 (R2014b)*, with $\varepsilon_M \approx 2.2 \cdot 10^{-16}$.

Let $x^* = A^{-1}b$ be the exact solution to $Ax = b$ and let $\widetilde{x}_k$ be the computed approximation to x^* by IR(ω). We produced the $n \times n$ matrix A and the vector $b = Ax^*$, with $x^* = [1,1,\ldots,1]^T$.

We report the following statistics for each iteration:

• forward stability error

$$\alpha(A,b,\tilde{x}_k) = \frac{\| \tilde{x}_k - x^* \|}{\kappa(A) \| x^* \|}, \tag{20}$$

• backward stability error

$$\beta(A,b,\tilde{x}_k) = \frac{\| b - A\tilde{x}_k \|}{\| A \| \| \tilde{x}_k \|}, \tag{21}$$

• componentwise backward stability error

$$\gamma(A,b,\tilde{x}_k) = \max_i \frac{(|b - A\tilde{x}_k|)_i}{(|A||\tilde{x}_k|)_i}. \tag{22}$$

Note that, the componentwise stability implies the backward stability, and backward stability implies forward stability.

We consider the following solvers S.

• **Algorithm I (GEPP).** Gaussian elimination with partial pivoting (GEPP) for the system $Ax = b$.

• **Algorithm II (BLU).** This method uses a block LU factorization [2]:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} I & 0 \\ L_{21} & I \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}. \quad (23)$$

We assume that $A_{11}(m \times m)$ is nonsingular. Then

(a) $U_{11} = A_{11}$, $U_{12} = A_{12}$.

(b) Solve the system $L_{21}A_{11} = A_{21}$ for L_{21} (by GEPP).

(c) Compute the Schur complement $U_{22} = A_{22} - L_{21}A_{12}$.

Next we solve the system $LUx = b$ by solving two linear systems, using the *MATLAB* commands

```
y=L\b; x=U\y;
```

Example 1 Take $A = W_n$, where W_n is the famous Wilkinson's matrix of order n:

$$W_n = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 1 \\ -1 & 1 & 0 & \dots & 0 & 1 \\ -1 & -1 & 1 & \dots & 0 & 1 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ -1 & -1 & -1 & \dots & -1 & 1 \end{bmatrix}. \quad (24)$$

R.D. Skeel [8] wrote: "Gaussian elimination with pivoting is not always as accurate as one might reasonably expect". It is known, see [10], that GEPP is considered numerically stable unless the growth factor ρ_n is large. For Wilkinson's matrix W_n we have $\rho_n = 2^{n-1}$. It is interesting that for $n = 100$ the Wilkinson matrix is perfectly well-conditioned, but GEPP produces an unstable solution! After one step of Wilkinson's iterative refinement method (for $\omega = 1$) we get the exact solution $x^* = [1,1,\dots,1]^T$. The situation is different for other choices of parameter ω. The results are contained in Table **1**.

Example 2 We test Algorithm I (GEPP) on badly scaled tridiagonally matrix A generated by the MATLAB code

```
randn('state',0)
n=10;m=5;
u=randn(n,1); v=randn(n-1,1);
A=diag(u)+diag(v,-1)+diag(v,1);
t=1e10; A(m-1,m)=t;
end
```

Random matrices of entries were generated by the *MATLAB* function "randn" (normally distributed pseudorandom numbers). Before each usage the random number generator was reset to its initial state. Notice that only the element $A_{4,5}$ is very large (equals 10^{10}), hence the matrix A is ill-conditioned. The values of the componentwise stability error (22) are gathered in Table **2**. Clearly the best results are obtained for $\omega = 1$ (Wilkinson's original iterative refinement). We don't display the forward error (20) and backward stability error (21) because they were always small (of order ε_M).

Example 3 We generate a block matrix A as in (23) using the following MATLAB code.

```
m=8; n=2*m;
rand('state',0);
A=rand(n);
A(1:m,1:m)=hilb(m);
```

Table 1: Values of the Forward Stability Error (20) for Algorithm I (GEPP) , where A is the 100×100 Wilkinson Matrix Defined in (24). Here $\kappa(A) = 44.8$

ω / k	0.3	0.5	0.7	0.9	1.0	1.2
0	1.51E-02	1.51E-02	1.51E-02	1.51E-02	1.51E-02	1.51E-02
1	1.05E-02	7.56E-03	4.54E-03	1.51E-03	0	3.02E-03
2	7.41E-03	3.78E-03	1.36E-03	1.51E-04	0	6.05E-04
3	5.19E-03	1.89E-03	4.08E-04	1.51E-05	0	1.21E-04
4	3.63E-03	9.46E-04	1.22E-04	1.51E-06	0	2.42E-05
5	2.54E-03	4.73E-04	3.67E-05	1.51E-07	0	4.84E-06
6	1.78E-03	2.36E-04	1.10E-05	1.51E-08	0	9.68E-07
7	1.24E-03	1.18E-04	3.31E-06	1.51E-09	0	1.93E-07
8	8.72E-04	5.91E-05	9.93E-07	1.51E-10	0	3.87E-08
9	6.10E-04	2.95E-05	2.97E-07	1.51E-11	0	7.75E-09
10	4.27E-04	1.47E-05	8.93E-08	1.51E-12	0	1.55E-09

Table 2: Values of the Componentwise Stability Error (22) for Algorithm I (GEPP) , where A is the 10×10 Tridiagonal Matrix Defined in Example 2 for $t=10^{10}$. Here $\kappa(A)=7.74\cdot 10^{10}$

ω / k	0.3	0.5	0.7	0.9	1.0	1.2
0	1.02E-06	1.024E-06	1.02E-06	1.02E-06	1.02E-06	1.02E-06
1	7.15E-07	5.10E-07	3.06E-07	1.02E-07	1.15E-16	2.04E-07
2	5.00E-07	2.55E-07	9.19E-08	1.02E-08	1.15E-16	4.08E-08
3	3.50E-07	1.27E-07	2.75E-08	1.02E-09	1.15E-16	8.17E-09
4	2.45E-07	6.38E-08	8.27E-09	1.02E-10	1.15E-16	1.63E-09
5	1.71E-07	3.19E-08	2.48E-09	1.02E-11	1.15E-16	3.27E-10
6	1.20E-07	1.59E-08	7.44E-10	1.021E-12	1.15E-16	6.54E-11
7	8.41E-08	7.98E-09	2.23E-10	1.021E-13	1.15E-16	1.30E-11
8	5.89E-08	3.99E-09	6.70E-11	1.01E-14	1.15E-16	2.61E-12
9	4.12E-08	1.99E-09	2.01E-11	1.07E-15	1.15E-16	5.23E-13
10	2.88E-08	9.97E-10	6.034E-12	1.54E-16	1.15E-16	1.04E-13

The matrix A is very well-conditioned, with the condition number $\kappa(A)=2.08\cdot 10^{2}$ but the block $(1,1)$ of A is ill-conditioned: $\kappa(A_{11})=4.75\cdot 10^{8}$. Here $H=hilb(m)$ is a $m\times m$ Hilbert matrix defined by

$$H=(h_{ij}),\, h_{ij}=\frac{1}{i+j-1},\, i,j=1,\dots,m.$$

The results are contained in Tables **3-3**.

Based on the numerical results of this section, we conclude that one step of Wilkinson's iterative

Table 3: Values of the Forward Stability Error (20) for Algorithm II (BLU), where A is the 16×16 Matrix Defined in Example 3

ω / k	0.3	0.5	0.7	0.9	1.0	1.2
0	2.01E-10	2.01E-10	2.01E-10	2.01E-10	2.01E-10	2.01E-10
1	1.41E-10	1.00E-10	6.05E-11	2.01E-11	3.57E-17	4.03E-11
2	9.89E-11	5.04E-11	1.81E-11	2.01E-12	2.64E-17	8.07E-12
3	6.92E-11	2.52E-11	5.45E-12	2.01E-13	9.57E-18	1.61E-12
4	4.84E-11	1.26E-11	1.63E-12	2.01E-14	8.91E-18	3.23E-13
5	3.39E-11	6.31E-12	4.90E-13	2.01E-15	1.19E-17	6.46E-14
6	2.37E-11	3.15E-12	1.47E-13	1.96E-16	2.94E-17	1.29E-14
7	1.66E-11	1.57E-12	4.41E-14	2.62E-17	1.46E-17	2.58E-15
8	1.16E-11	7.88E-13	1.32E-14	3.71E-17	2.04E-17	5.12E-16
9	8.14E-12	3.94E-13	3.96E-15	5.36E-17	2.47E-17	9.83E-17
10	5.70E-12	1.97E-13	1.19E-15	2.70E-17	3.22E-17	4.84E-17

Table 4: Values of the Backward Stability Error (21) for Algorithm II (BLU), where A is the 16×16 Matrix Defined in Example 3

ω / k	0.3	0.5	0.7	0.9	1.0	1.2
0	4.03E-09	4.03E-09	4.03E-09	4.03E-09	4.03E-09	4.03E-09
1	2.82E-09	2.01E-09	1.21E-09	4.03E-10	1.90E-16	8.06E-10
2	1.97E-09	1.00E-09	3.63E-10	4.03E-11	1.92E-16	1.61E-10
3	1.38E-09	5.04E-10	1.08E-10	4.03E-12	1.43E-16	3.22E-11
4	9.68E-10	2.52E-10	3.26E-11	4.03E-13	1.53E-16	6.45E-12
5	6.78E-10	1.26E-10	9.80E-12	4.03E-14	1.44E-16	1.29E-12
6	4.74E-10	6.30E-11	2.94E-12	4.01E-15	1.49E-16	2.58E-13
7	3.32E-10	3.15E-11	8.82E-13	4.14E-16	1.63E-16	5.16E-14
8	2.32E-10	1.575E-11	2.64E-13	1.14E-16	1.22E-16	1.03E-14
9	1.62E-10	7.88E-12	7.93E-14	8.18E-17	1.66E-16	2.09E-15
10	1.13E-10	3.94E-12	2.38E-14	1.44E-16	1.66E-16	5.16E-16

Table 5: Values of the Componentwise Backward Stability Error (22) for Algorithm II (BLU), where A is the 16×16 Matrix Defined in Example 3

ω / k	0.3	0.5	0.7	0.9	1.0	1.2
0	7.88E-09	7.88E-09	7.88E-09	7.88E-09	7.88E-09	7.88E-09
1	5.51E-09	3.94E-09	2.36E-09	7.88E-10	4.19E-16	1.57E-09
2	3.86E-09	1.97E-09	7.09E-10	7.88E-11	4.61E-16	3.15E-10
3	2.70E-09	9.85E-10	2.12E-10	7.88E-12	3.07E-16	6.30E-11
4	1.89E-09	4.92E-10	6.38E-11	7.88E-13	3.07E-16	1.26E-11
5	1.32E-09	2.46E-10	1.91E-11	7.89E-14	2.79E-16	2.52E-12
6	9.27E-10	1.23E-10	5.74E-12	7.87E-15	2.27E-16	5.04E-13
7	6.49E-10	6.15E-11	1.72E-12	7.95E-16	3.07E-16	1.01E-13
8	4.54E-10	3.07E-11	5.17E-13	2.25E-16	2.21E-16	2.03E-14
9	3.18E-10	1.53E-11	1.54E-13	1.89E-16	3.04E-16	4.15E-15
10	2.22E-10	7.69E-12	4.65E-14	3.78E-16	3.41E-16	1.02E-15

refinement method ($\omega = 1$) is usually be enough to yield small errors (20)–(22). However, iterative refinement method with a relaxation ω which is not close to 1, can require much more steps than Wilkinson's iterative refinement. Therefore, the choice $\omega = 1$ is the best choice from the point of numerical stability.

REFERENCES

[1] Buttari A, Dongarra J, Langou J, Langou J, Luszczek JP, Kurzak J. Mixed precision iterative refinement techniques for the solution of dense linear systems. International Journal of High Performance Computing Applications 2007; 21(4): 457-466. https://doi.org/10.1177/1094342007084026

[2] Demmel JW, Higham NJ, Schreiber RS. Stability of block LU factorization. Numer Linear Algebra Appl 1995; 12: 173-190. https://doi.org/10.1002/nla.1680020208

[3] Foster LV. Gaussian elimination with partial pivoting can fail in practice. SIAM J Matrix Anal Appl 1994; 15(4): 1354-1362. https://doi.org/10.1137/S0895479892239755

[4] Higham NJ. Iterative refinement enhances the stability of QR factorization methods for solving linear equations. BIT 1991; 31: 447-468. https://doi.org/10.1007/BF01933262

[5] Higham NJ. Iterative refinement for linear systems and LAPACK. IMA J Numer Anal 1997; 17: 495-509. https://doi.org/10.1093/imanum/17.4.495

[6] Jankowski M, Woźniakowski H. Iterative refinement implies numerical stability. BIT 1977; 17: 303-311. https://doi.org/10.1007/BF01932150

[7] Rozložník M, Smoktunowicz A, Kopal J. A note on iterative refinement for seminormal equations. Applied Numerical Mathematics 2014; 75: 167-174. https://doi.org/10.1016/j.apnum.2013.08.005

[8] Skeel RD. Iterative refinement implies numerical stability for Gaussian elimination. Math Comp 1980; 35: 817-832. https://doi.org/10.1090/S0025-5718-1980-0572859-4

[9] Smoktunowicz A, Smoktunowicz A. Iterative refinement techniques for solving block linear systems of equations. Applied Numerical Mathematics 2013; 67: 220-229. https://doi.org/10.1016/j.apnum.2011.11.004

[10] Wilkinson JH. The Algebraic Eigenvalue Problem, Oxford University Press 1965.

[11] Wu X, Wang Z. A new iterative refinement with roundoff error analysis. Numer Linear Algebra Appl 2011; 18: 275-282. https://doi.org/10.1002/nla.723

Received on 22-11-2016 Accepted on 25-12-2016 Published on 30-12-2016

DOI: http://dx.doi.org/10.15377/2409-5761.2016.03.02.1