Numerical stability of iterative refinement with a relaxation for linear systems

Alicja Smoktunowicz, Jakub Kierzkowski and Iwona Wróbel

December 15, 2015

Abstract

Stability analysis of Wilkinson’s iterative refinement with a relaxation IR(\(\omega\)) for solving linear systems is given. It extends existing results for \(\omega = 1\), i.e., for Wilkinson’s iterative refinement. 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 with a relaxation. Our tests confirm that the choice \(\omega = 1\) is the best choice from the point of numerical stability.

Keywords: Iterative refinement, numerical stability, condition number

AMS2010 MSC: 65F05, 65F10, 15A12

1 Introduction

We consider the system \(Ax = b\), where \(A \in \mathbb{R}^{n \times n}\) is nonsingular and \(b \in \mathbb{R}^{n}\). Iterative refinement techniques for linear systems of equations are very useful in practice and the literature on this subject is very rich (see [1]–[5], [8]–[18]).

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 with a relaxation IR(\(\omega\)) consists of three steps.
Algorithm IR(ω)

Given ω > 0. Let $x_0$ be computed by the solver $S$.
For $k = 0, 1, 2, \ldots$, the $k$th 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 (see [16]). Wu and Wang (see [17]) 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 [17] 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 \( x_{k+1} = x^* + \omega (x^* - x_k) \), \( k = 0, 1, \ldots \), 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 \( \omega \) 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 (see Chapter 2 in [10]). For two floating point numbers \( a \) and \( b \) we have

\[
\text{fℓ}(a \circ b) = (a \circ b)(1 + \Delta), \quad |\Delta| \leq \varepsilon_M
\]

for results in the normalized range, where \( \circ \) denotes any of the elementary scalar operations +, -, *, / and \( \varepsilon_M \) is machine precision.

In this paper we present a comparison of Wilkinson’s iterative refinement 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

\[
\|\tilde{x} - x^*\| \leq O(\varepsilon_M)\kappa(A)\|x^*\|. \tag{1}
\]
Throughout this paper, $\| \cdot \|$ is the matrix or vector two–norm depending on 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 $\tilde{x}$ in floating point arithmetic is the exact solution of a slightly perturbed system

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

Our analysis is similar in spirit to [8]–[11] and [14]–[15]. Jankowski and Woźniakowski proved in [11] 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($\omega$).

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

## 2 Forward stability of IR($\omega$)

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

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

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

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

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

We start with the following lemma.

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

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

and

$$|1 - \omega| + \omega q \leq 0.6. \quad (6)$$
Then for $k = 0, 1, \ldots$ we have

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

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. $\|\tilde{x}_{k+1} - x^*\| \leq q_{k+1} \|x^*\|$, where $q_{k+1} \leq 0.1$ and $q_{k+1}$ satisfies (8).

Under assumption (11), the computed vectors $\tilde{r}_k$ satisfy

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

Under assumption (3) we have

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

Standard error analysis shows

$$
\tilde{x}_{k+1} = (I + D^{(1)}_k)(\tilde{x}_k + (I + D^{(2)}_k)\omega \tilde{p}_k), \quad \|D^{(i)}_k\| \leq \varepsilon M. \quad (11)
$$

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

$$
\|\tilde{x}_k\| = \|x^* + (\tilde{x}_k - x^*)\| \leq \|x^*\| + \|\tilde{x}_k - x^*\| \leq (1 + q_k)\|x^*\|. \quad (12)
$$

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

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

From (9) and the inequality $\|b\| = \|Ax^*\| \leq ||A||\|x^*\|$ it can be seen that

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

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

$$
\|p_k^*\| \leq \|\tilde{x}_k - x^*\| + \|\xi_k\|, \quad \|\xi_k\| \leq 2.1L(n)\varepsilon M \kappa(A)\|x^*\|. \quad (15)
$$
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, \(\|D_k^{(3)}\| \leq 2\varepsilon_M + \varepsilon_M^2\), so from (11) we get
\[
\tilde{x}_{k+1} = (\tilde{x}_k + \omega \tilde{p}_k) + \eta_k, \quad \|\eta_k\| \leq \varepsilon_M \|\tilde{x}_k\| + (2\varepsilon_M + \varepsilon_M^2)\omega \|\tilde{p}_k\|.
\] 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
\[
\|\tilde{x}_{k+1} - x^*\| \leq |1 - \omega|\|\tilde{x}_k - x^*\| + \|\eta_k\| + \omega\|\xi_k\| + \omega q\|p_k^*\|.
\] (17)
First we estimate \(\|\eta_k\|\). Since \(\|\tilde{x}_k - x^*\| \leq 0.1\|x^*\|\), so by assumption (5) we obtain from (15) the bounds
\[
\|\xi_k\| \leq 0.021\|x^*\|, \quad \|p_k^*\| \leq 0.121\|x^*\|.
\] (18)
From (12) and (16) we have \(\|\eta_k\| \leq 1.1\varepsilon_M(\|x^*\| + (2 + \varepsilon_M)\omega \|p_k^*\|)\). Now we apply (5) and (18). Since \(\omega < 2\), we see that \(\|\eta_k\| \leq 1.64\varepsilon_M\|x^*\|\).
Therefore,
\[
\omega\|\xi_k\| + \|\eta_k\| \leq \omega 2.1L(n)\varepsilon_M\kappa(A)\|x^*\| + 1.64\varepsilon_M\|x^*\|
\]
and by (15) we get
\[
\omega q\|p_k^*\| \leq \omega q\|\tilde{x}_k - x^*\| + \omega q 2.1L(n)\varepsilon_M\kappa(A)\|x^*\|.
\]
Hence, from (17) and by (5)-(6) we finally obtain
\[
\|\tilde{x}_{k+1} - x^*\| \leq (|1 - \omega| + \omega q)\|\tilde{x}_k - x^*\| + 2.31\omega L(n)\varepsilon_M\kappa(A) + 1.64\varepsilon_M\|x^*\|.
\]
We conclude that \(\|\tilde{x}_{k+1} - x^*\| \leq q_{k+1}\|x^*\|\), with \(q_{k+1}\) defined in (8). It remains to prove that \(q_{k+1} \leq 0.1\). By assumptions (5) and (6) and using the fact that \(q_k \leq 0.1\), we see that \(q_{k+1} \leq 0.6 \ast 0.1 + (0.0231 + 0.0164)\), so \(q_{k+1} \leq 0.1\). This completes the proof. \(\blacksquare\)

**Theorem 2.1** Under the assumptions of Lemma 2.1 the algorithm \(IR(\omega)\) is forward stable for \(\omega \in (0, 2)\). There exists \(k^*\) depending only on \(n\) such that for every \(k \geq k^*\) the following inequality holds
\[
\|\tilde{x}_k - x^*\| \leq (11.6L(n) + 4.2)\varepsilon_M\kappa(A)\|x^*\|.
\] (19)
Proof. We apply the results of Lemma 2.1. Notice that from (7)-(8) and by assumptions (5) it follows that
\[ q_{k+1} \leq q_{k} + 2.31 \omega L(n) \varepsilon_M \kappa(A) + 1.64 \varepsilon_M. \]
Since \( \omega < 2 \) and \( 1 \leq \kappa(A) \), we get
\[ q_{k+1} \leq q_{k} + (4.62L(n) + 1.64)\varepsilon_M \kappa(A). \]
From this it follows that
\[ q_{k+1} \leq (0.6)^k + \frac{4.62L(n) + 1.64}{1 - 0.6} \varepsilon_M \kappa(A). \]
From this (19) follows immediately. \( \square \)

3 Numerical tests

In this section we present numerical experiments that show the comparison of the IR(\( \omega \)) for different values of \( \omega \). 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 \( \tilde{x}_k \) be the computed approximation to \( x^* \) by IR(\( \omega \)). 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**
  \[ \alpha(A, b, \tilde{x}_k) = \frac{\| \tilde{x}_k - x^* \|}{\kappa(A) \| x^* \|}, \quad (20) \]

- **backward stability error**
  \[ \beta(A, b, \tilde{x}_k) = \frac{\| b - A\tilde{x}_k \|}{\| A \| \| \tilde{x}_k \|}, \quad (21) \]

- **componentwise backward stability error**
  \[ \gamma(A, b, \tilde{x}_k) = \max_i \left( \frac{|b - A\tilde{x}_k|}{|A| |\tilde{x}_k|} \right)_i. \quad (22) \]

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 $A = LU$ (6):

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

We assume that $A_{11}$ is nonsingular. Then

1. $U_{11} = A_{11}$, $U_{12} = A_{12}$.
2. Solve the system $L_{21}A_{11} = A_{21}$ for $L_{21}$ (by GEPP).
3. 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

$y = L \backslash b; \quad x = U \backslash y$;

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

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

We cite R.D. Skeel who wrote in [14]: ”Gaussian elimination with pivoting is not always as accurate as one might reasonably expect”. It is known (see [14]) that GEPP is considered numerically stable unless the growth factor $\rho_n$ is large. For Wilkinson’s matrix $W_n$ we have $\rho_n = 2^{n-1}$. Description of other types of matrices for which the growth factor is very large is given in [3] and [10], Section 9. 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 (for $\omega = 1$) we get the exact solution $x^* = (1, 1, \ldots, 1)^T$. The situation is different for other choices of parameter $\omega$. 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
Table 1: Values of the forward stability error (20) for Algorithm I (GEPP), where $A$ is the $100 \times 100$ Wilkinson matrix defined in (24). Here $\kappa(A) = 44.8$.

| $\omega/k$ | 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 |

```
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 from the distribution $N(0,1)$. They were generated by the MATLAB function "randn". 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 $\varepsilon M$).

**Example 3** We generate a block matrix $A$ as in (24) using the following MATLAB code.

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

8
Table 2: Values of the componentwise stability error (22) for Algorithm I (GEPP), where $A$ is the $10 \times 10$ tridiagonal matrix defined in Example 2 for $t = 10^{10}$. Here $\kappa(A) = 7.74 \cdot 10^7$.

| $\omega/k$ | 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  | 1.02E-12  | 1.15E-16  | 6.54E-11  |
| 7          | 8.41E-08  | 7.98E-09  | 2.23E-10  | 1.02E-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 $H = \text{hilb}(m)$ is a $m \times m$ Hilbert matrix defined by

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

The results are contained in Tables 3-5.

Based on the numerical results of this section, we conclude that one step of Wilkinson’s iterative refinement ($\omega = 1$) is usually be enough to yield small errors (20), (22). However, iterative refinement with a relaxation $\omega$ 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] M. Arioli, J. Demmel, I.S. Duff, Solving sparse linear systems with sparse backward error, SIAM J. Matrix Anal. Appl. 10 (1989), 165–190.

[2] M. Arioli, J.Scott, Chebyshev acceleration of iterative refinement, Numer. Algor. 66 (2014), 591-608.
Table 3: Values of the forward stability error (20) for Algorithm II (BLU), where $A$ is the $16 \times 16$ matrix defined in Example 3.

| $\omega/k$ | 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 $16 \times 16$ matrix defined in Example 3.

| $\omega/k$ | 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 \(\times\) 16 matrix defined in Example 3.

| \(\omega/k\) | 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 | 4.19E-16 | 4.19E-16 | 4.19E-16 |
| 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 |

[3] Å. Björck, Iterative refinement and realiable computing. In Reliable Numerical Computation, M. G. Cox and S. J. Hammarling, editors, Oxford University Press (1990), 249–266.

[4] Å. Björck, Iterative refinement of linear least squares solutions I, BIT 7 (1967), 257–278.

[5] Å. Björck, Iterative refinement of linear least squares solutions II, BIT 8 (1968), 8–30.

[6] J.W. Demmel, N.J. Higham, and R.S. Schreiber, Stability of block LU factorization, Numer. Linear Algebra Appl. 12 (1995), 173–190.

[7] L.V. Foster, Gaussian elimination with partial pivoting can fail in practice, SIAM J. Matrix Anal. Appl. 15 (4) (1994), 1354–1362.

[8] N.J. Higham, Iterative refinement enhances the stability of QR factorization methods for solving linear equations, BIT 31 (1991), 447–468.

[9] N.J. Higham, Iterative refinement for linear systems and LAPACK, IMA J. Numer. Anal., 17(4) (1997), 495–509.

[10] N.J. Higham, Accuracy and Stability of Numerical Algorithms, Second Edition, SIAM, Philadelphia (2002).

[11] M. Jankowski, H. Woźniakowski, Iterative refinement implies numerical stability, BIT 17 (1977), 303–311.
[12] C.B.Moler, Iterative refinement in floating point, J.Assoc. Comput. Mach. 14 (2) (1967), 316–321.

[13] M.Rozložník, A.Smoktunowicz and J.Kopal, A note on iterative refinement for seminormal equations, Applied Numerical Mathematics 75 (2014), 167–174.

[14] R.D.Skeel, Iterative refinement implies numerical stability for Gaussian elimination, Math. Comp. 35 (1980), 817–832.

[15] Alicja Smoktunowicz and Agata Smoktunowicz, Iterative refinement techniques for solving block linear systems of equations, Applied Numerical Mathematics 67 (2013), 220-229.

[16] J. H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press (1965).

[17] X.Wu and Z.Wang, A new iterative refinement with roundoff error analysis, Numer. Linear Algebra Appl. 18 (2011), 275–282.

[18] Z. Zlatev, Use of iterative refinement in the solution of sparse linear systems, SIAM J. Numer. Anal., 19(2) (1982), 381-399.