Iterative refinement techniques for solving block linear systems of equations

Citation for published version:
Smoktunowicz, A & Smoktunowicz, A 2013, 'Iterative refinement techniques for solving block linear systems of equations' Applied Numerical Mathematics, vol. 67, pp. 220-229. DOI: 10.1016/j.apnum.2011.11.004

Digital Object Identifier (DOI):
10.1016/j.apnum.2011.11.004

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Applied Numerical Mathematics

Iterative refinement techniques for solving block linear systems of equations

Alicja Smoktunowicz[a,*], Agata Smoktunowicz[b,c]

[a] *Faculty of Mathematics and Information Science, Warsaw University of Technology, 00-661 Warsaw, Plac Politechniki 1, Poland*
[b] *School of Mathematics, University of Edinburgh, Edinburgh, Scotland EH9 3JZ, UK*
[c] *Institute of Mathematics of the Polish Academy of Sciences, ul. Sniadeckich 8, P.O. Box 21, 00-956 Warsaw, Poland*

Abstract

We study numerical properties of classical iterative refinement (IR) and k-fold iterative refinement (RIR) of solutions of a nonsingular linear system of equations $Ax = b$, with A partitioned into blocks, using only single precision. We prove that RIR has better numerical quality than IR.

Keywords: Iterative refinement; linear systems; block matrices; condition number; numerical stability.

2000 MSC: 65F10, 65G50, 15A12

1. Introduction

In many practical applications, e.g. arising in solving differential equations numerically, we need to solve a linear system of equations $Ax = b$, where

*Corresponding author

Email addresses: smok@mini.pw.edu.pl (Alicja Smoktunowicz), A.Smoktunowicz@ed.ac.uk (Agata Smoktunowicz)

URL: http://www.maths.ed.ac.uk/people/show/person/67 (Agata Smoktunowicz)

$A \in \mathbb{R}^{N,N}$ is nonsingular and has a special block structure. We assume that the matrix $A \in \mathbb{R}^{N,N}$ is partitioned into $s \times s$ blocks, i.e. $A = (A_{ij})$, where $A_{i,j} \in \mathbb{R}^{n_i,n_j}$ is refereed to as the (i,j) block of A, $\{n_1, \ldots, n_s\}$ is a given set of positive integers, $n_1 + \ldots + n_s = N$.

Very often, the block matrices A_{ij} are sparse and many of them are zero. Numerical algorithms ought to exploit the structure of the matrix A. We would like to use algorithms that produce solutions y accurate to full machine precision. Such algorithms are attractive because they preserve the structure of the matrix. If y solves a problem that is close to the original one, i.e.

$$(A+E)y = b, \ \|E_{ij}\|_2 \leq \epsilon \ \|A_{ij}\|_2, \ i,j = 1, \ldots, s,$$

then $A+E$ has the same block structure as A: $A_{ij} = 0$ implies that $E_{ij} = 0$.

If $A = (A_{ij})$ is symmetric then it is reasonable to have a numerical solution y as a solution of slightly perturbed symmetric system $(A+F)y = b$. We partly resolved this problem by using the blockwise approach (cf. [13]-[15]). If $A = (A_{ij}) \in \mathbb{R}^{N,N}$ is a block symmetric matrix and y is a solution of a nearby linear system $(A+E)y = b$, then there exists $F = F^T$ such that y solves a nearby symmetric system $(A+F)y = b$, if A is symmetric positive definite or the matrix $\mu(A)$ is diagonally dominant, or $\mu(A)$ is H-matrix, where $\mu(A)$ is a matricial norm of A (cf. [7], [11], [15]),

$$\mu(A) = \begin{pmatrix} \|A_{11}\|_2 & \|A_{12}\|_2 & \cdots & \|A_{1s}\|_2 \\ \|A_{21}\|_2 & \|A_{22}\|_2 & \cdots & \|A_{2s}\|_2 \\ \cdots & \cdots & \cdots & \cdots \\ \|A_{s1}\|_2 & \|A_{s2}\|_2 & \cdots & \|A_{ss}\|_2 \end{pmatrix}. \quad (1)$$

Without loss of generality we restrict our attention to the spectral matrix norm (2-norm) and the second vector norm (length of x). It is well-known that $\|A\|_2^2 = \rho(A^TA)$, where $\rho(A) = \max\{|\lambda| : \lambda \in \sigma(A)\}$ denotes the spectral radius of A.

If the vector $x \in \mathbb{R}^N$ is partitioned as $x = ({x_1}^T, \ldots, {x_s}^T)^T$ where $x_i \in \mathbb{R}^{n_i}$, then $\mu(x) = (\|x_1\|_2, \ldots, \|x_s\|_2)^T$ and $\|x\|_2 = \|\mu(x)\|_2$.

Matricial norms have very elegant properties (cf. [7], [11], [13], [15]), for example, for block matrices $A = (A_{ij})$, $B = (B_{ij})$ and vectors x, y, partitioned conformally, we have

- $\mu(A + B) \leq \mu(A) + \mu(B), \quad \mu(AB) \leq \mu(A)\mu(B),$
- $\mu(Ax) \leq \mu(A)\mu(x), \quad \|Ax\|_2 \leq \|\mu(A)\mu(x)\|_2,$
- $\rho(A) \leq \rho(\mu(A))$ (the Frobenius inequality),
- $\|A\|_2 \leq \|\mu(A)\|_2.$

Here inequalities between matrices $A = (A_{ij})$ and $B = (B_{ij})$ are understood to hold for all blocks A_{ij} and B_{ij}, i.e. $\mu(A) \leq \mu(B)$ means that for all i, j we have $\|A_{ij}\|_2 \leq \|B_{ij}\|_2$.

Our blockwise analysis extend existing normwise and componentwise results on preserving symmetric perturbations. Some important cases are: $\mu(A) = |A| = (|a_{ij}|)$ for $s = n$ (componentwise case) and $\mu(A) = \|A\|_2$ for $s = 1$ (normwise case).

We can measure the sensitivity of the solution of linear system $Ax = b$ with respect to the perturbations of the blocks A_{ij}. Notice that if x^* is the exact solution to $Ax = b$ and $\hat{x}$ is the exact solution to a slightly perturbed system $(A + \Delta A)\hat{x} = b$ with $\mu(\Delta A) \leq \epsilon\mu(A)$ then $\hat{x} - x^* = -A^{-1}\Delta A\hat{x}$, so

$$\mu(\tilde{x} - x^*) \leq \epsilon \; \mu(A^{-1})\mu(A)\mu(x^*) + \mathcal{O}(\varepsilon_M^2).$$

From this it follows that

$$\|\hat{x} - x^*\|_2 \leq \epsilon \|\mu(A^{-1})\mu(A)\mu(x^*)\|_2 + \mathcal{O}(\varepsilon_M^2).$$

Let

$$\Omega = \mu(A^{-1})\mu(A), \;\; \kappa_\mu(A) = \|\Omega\|_2. \tag{2}$$

We call $\kappa_\mu(A)$ the blockwise condition number of A and

$$cond_\mu(A; x^*) = \|\Omega \;\mu(x^*)\|_2 / \|x^*\|_2 \tag{3}$$

the blockwise condition number of the nonzero solution x^* to the system $Ax = b$ (cf. [13]).

The blockwise condition number measures the sensitivity of the system $Ax = b$ with respect to the partition $\{n_1, \ldots, n_s\}$. It is easy to see that

$$1 \leq cond_\mu(A; x^*) \leq \kappa_\mu(A) \leq s^2 \;\kappa(A),$$

where $\kappa(A) = \|A^{-1}\|_2 \;\|A\|_2$ is the normwise condition number of A.

Now it is natural to introduce blockwise stability of algorithms in the solving of linear system of equations.

Definition 1.1. An algorithm for solving a block system $Ax = b$ with nonsingular A partitioned into blocks A_{ij} is *strongly blockwise forward stable* if the computed solution $\tilde{x}$ in floating point arithmetic (fl) satisfies

$$\mu(\tilde{x} - x^*) \leq L_1 \varepsilon_M \;\Omega \;\mu(x^*) + \mathcal{O}(\varepsilon_M^2), \tag{4}$$

where $L_1 = L_1(N)$ is a modestly growing function of N, ε_M is machine precision and x^* is the exact solution to $Ax = b$.

An algorithm for solving a block system $Ax = b$ is *blockwise forward stable* if the computed solution $\tilde{x}$ satisfies

$$\frac{\|\tilde{x} - x^*\|_2}{\|x^*\|_2} \leq L_2 \varepsilon_M \ cond_\mu(A; x^*), \tag{5}$$

where $L_2 = L_2(N)$ is a modestly growing function of N.

An algorithm for solving a block system $Ax = b$ is *strongly blockwise backward stable* if the computed solution $\tilde{x}$ satisfies

$$(A + \Delta A)\tilde{x} = b, \ \ \mu(\Delta A) \leq L_3 \varepsilon_M \mu(A), \tag{6}$$

where $L_3 = L_3(N)$ is a modestly growing function of N.

An algorithm for solving a block system $Ax = b$ is *blockwise backward stable* if the computed solution $\tilde{x}$ satisfies

$$\|b - A\tilde{x}\|_2 \leq L_4 \varepsilon_M \ \|\mu(A) \ \mu(\tilde{x})\|_2, \tag{7}$$

where $L_4 = L_4(N)$ is a modestly growing function of N.

Rigal and Gaches (cf. [17]) prove that (6) is equivalent to the following condition

$$\mu(b - A\tilde{x}) \leq L_3 \varepsilon_M \mu(A) \ \mu(\tilde{x}). \tag{8}$$

Clearly, strong blockwise stability implies blockwise stability and blockwise backward stability implies blockwise forward stability. In componentwise analysis, our definition of blockwise backward stability is the same as R-stability introduced by Skeel ([12]). In this paper we focus our attention only on blockwise forward stability. Full consideration of other measures of numerical stability of algorithms for solving block linear systems of equations exceeds the scope of this paper.

We consider some ways in which iterative refinement may be used to improve the computed results. Several papers and reports on iterative refinement have appeared (cf. [1]-[6]).

We present various kinds of iterative refinement techniques, e.g. k-fold iterative refinement, for the solution of a nonsingular system $Ax = b$ with A partitioned into blocks using only single precision arithmetic. Iterative refinement may give solutions to full single precision even when the initial solution has no correct significant figures. Very often, one or two steps are sufficient to terminate the process successfully. Numerical tests were done in MATLAB to compare the performance of some direct methods for solving linear system of equations of special block matrices.

2. Algorithms

We investigate some computational aspects of **Recurrent Iterative Refinement (RIR)** for linear system of equations $Ax = b$, where $A \in \mathbb{R}^{N,N}$ is partitioned into $s \times s$ blocks, i.e. $A = (A_{ij})$ with $A_{i,j} \in \mathbb{R}^{n_i,n_j}$. RIR (k-fold iterative refinement) is a generalization of the well-known classical **Iterative Refinement (IR)** technique for improving the accuracy of weakly stable algorithms for solving linear system of equations. Recurrent iterative refinement was proposed by Woźniakowski (cf. [9], [13]).

The idea of Recurrent Iterative Refinement is to decompose first the matrix A to factors of simple structure (e.g. triangular, orthogonal, bidiagonal, diagonal, block LU, block Q-R etc.) and then use iterative refinement techniques to correct a computed solution x_0 by a solver $S_0(b)$ (i.e. $x_0 = S_0(b)$). $S_0(b)$ solves in floating point arithmetic (fl) the linear system $Ax = b$ using the given decomposition of A.

A single iteration of iterative refinement in floating point arithmetic (fl) is given by 1-fold iterative refinement as follows:

$$x_1 = S_1(b) \Longleftrightarrow \begin{cases} x_0 = S_0(b) \\ r_0 = b - Ax_0 \\ p_0 = S_0(r_0) \\ x_1 = x_0 + p_0. \end{cases} \tag{9}$$

Notice that "in theory" x_1 will be the exact solution x^* to $Ax = b$ but hardly ever in floating point arithmetic.

If we replace S_0 in (1) by S_k then we define $(k+1)$-fold iterative refinement. Thus $x_{k+1} = S_{k+1}(b)$ is as follows:

$$x_{k+1} = S_{k+1}(b) \quad \Longleftrightarrow \begin{cases} x_k = S_k(b) \\ r_k = b - Ax_k \\ p_k = S_k(r_k) \\ x_{k+1} = x_k + p_k. \end{cases} \tag{10}$$

If $p_k = S_k(r_k)$ in (10) is replaced by $p_k = S_0(r_k)$ then this method is k iterations of classical **Iterative Refinement (IR)**.

We see that k-fold iterative refinement requires additional storage proportional to the depth of the recursion which is not so large.

3. Blockwise stability

Let $x^* = A^{-1}b$ is the exact solution to $Ax = b$. We need a basic (direct or iterative) linear equation solver S_0 for $Ax = b$ such that

$$\|S_0(b) - x^*\|_2 \leq q_0 \ \|x^*\|_2, \ q_0 \leq 0.1. \tag{11}$$

This condition can be replaced by the assumption that $q_0 < 1$ and q_0 is not too close to unity. We use (11) to simplify error analysis.

Next, we assume that the matrix-vector multiplication is blockwise backward stable, i.e. there exists a matrix E such that

$$fl(Ax) = (A+E)x, \;\; \mu(E) \leq L\varepsilon_M \mu(A), \;\;\; L \geq 1, \tag{12}$$

where $L = L(N)$ is a small constant depending only on N.

Lemma 3.1. *Let k-fold iterative refinement be applied to the nonsingular block linear system $Ax = b$, using the solver S_0 satisfying (11)-(12). Let $x_k = S_k(b)$ denote the computed vectors in floating point arithmetic. Assume that*

$$\varepsilon_M \leq 0.01, \;\; L\varepsilon_M \, \kappa_\mu(A) \leq 0.01. \tag{13}$$

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

$$\|x_k - x^*\|_2 \leq q_k \; \|x^*\|_2, \;\; q_k \leq 0.1, \tag{14}$$

where

$$q_{k+1} = q_k^2 + 2.5L\varepsilon_M \; (cond_\mu(A; x^*) + q_k\kappa_\mu(A)). \tag{15}$$

Proof. Assume that (14) holds for k. We prove that it holds also for $k+1$, i.e. $\|x_{k+1} - x^*\|_2 \leq q_{k+1} \; \|x^*\|_2$, where $q_{k+1} \leq 0.1$ and q_{k+1} satisfies (15).

The computed vectors r_k, p_k and x_{k+1} in floating point arithmetic by k-fold iterative refinement S_k satisfy

$$\begin{cases} x_k = x^* + \Delta x_k, \;\; x^* = A^{-1}b, \;\; \|\Delta x_k\|_2 \leq q_k \; \|x^*\|_2, \\ r_k = (I + D_k)(b - (A + E_k)x_k), \\ p_k = p_k^* + \Delta p_k, \;\; p_k^* = A^{-1}r_k, \;\; \|\Delta p_k\|_2 \leq q_k \; \|p_k^*\|_2, \\ x_{k+1} = (I + G_k)x_{k+1}^*, \;\; x_{k+1}^* = x_k + p_k, \end{cases} \tag{16}$$

where

$$\mu(E_k) \le L\varepsilon_M \mu(A), \;\; \mu(D_k) \le \varepsilon_M I, \;\; \mu(G_k) \le \varepsilon_M I. \tag{17}$$

Notice that

$$\Delta x_{k+1} = x_{k+1} - x^* = (I + G_k)(x^*_{k+1} - x^*) + G_k x^*,$$

so

$$\|\Delta x_{k+1}\|_2 \le (1 + \varepsilon_M)\|x^*_{k+1} - x^*\|_2 + \varepsilon_M \|x^*\|_2. \tag{18}$$

Now we would like to estimate $\|x^*_{k+1} - x^*\|_2$. We have

$$x^*_{k+1} - x^* = (x_k - x^*) + p^*_k + \Delta p_k.$$

After easy algebraic manipulations we obtain

$$\begin{cases} p^*_k = (x^* - x_k) - (\xi_k + \eta_k), \\ x^*_{k+1} - x^* = \Delta p_k - (\xi_k + \eta_k), \\ \xi_k = A^{-1}(I + D_k)E_k \; x^*, \\ \eta_k = A^{-1}((I + D_k)E_k + D_k A) \; \Delta x_k. \end{cases} \tag{19}$$

We see that

$$\mu(\xi_k) \le L\varepsilon_M (1 + \varepsilon_M)\Omega \; \mu(x^*)$$

and

$$\mu(\eta_k) \le (1 + L(1 + \varepsilon_M))\varepsilon_M \Omega \; \mu(\Delta x_k).$$

Applying (13), taking norms and using the assumption $\|\Delta x_k\|_2 \le q_k \|x^*\|_2$ we get

$$\|\xi_k\|_2 \le 1.01 L\varepsilon_M \|\Omega \; \mu(x^*)\|_2. \tag{20}$$

Since $1 \le L$ and $\varepsilon_M \le 0.01$ we obtain

$$\|\eta_k\|_2 \le 2.01 L q_k \varepsilon_M \kappa_\mu(A) \; \|x^*\|_2. \tag{21}$$

From (16) and (19) we get

$$\|x_{k+1}^* - x^*\|_2 \leq \|\Delta p_k\|_2 + \|\xi_k\|_2 + \|\eta_k\|_2$$

and

$$\|p_k^*\|_2 \leq \|\Delta x_k\|_2 + \|\xi_k\|_2 + \|\eta_k\|_2.$$

Since $\|\Delta x_k\|_2 \leq q_k \|x^*\|_2$ and $\|\Delta p_k\|_2 \leq q_k \|p_k^*\|_2$, we obtain

$$\|x_{k+1}^* - x^*\|_2 \leq q_k^2 \|x^*\|_2 + (1 + q_k)(\|\xi_k\|_2 + \|\eta_k\|_2).$$

By assumption, $q_k \leq 0.1$, hence from (20)- (21) we get

$$\|x_{k+1}^* - x^*\|_2 \leq q_k^2 \|x^*\|_2 + 1.2L\varepsilon_M(\|\Omega\ \mu(x^*)\|_2 + 2q_k\kappa_\mu(A)\ \|x^*\|_2).$$

From this, (18), (13) and the inequality $\|x^*\|_2 \leq \|\Omega\ \mu(x^*)\|_2$ it follows that

$$\|x_{k+1} - x^*\|_2 \leq q_k^2 \|x^*\|_2 + 2.5L\varepsilon_M(\|\Omega\ \mu(x^*)\|_2 + q_k\kappa_\mu(A)\ \|x^*\|_2).$$

Dividing this equation by $\|x^*\|_2$ and using (2)-(3) we see that $\|x_{k+1} - x^*\|_2 \leq q_{k+1}\|x^*\|_2$, with q_{k+1} given by (15). Notice that $q_{k+1} \leq (0.1)^2 + 0.025 + 0.025$, so $q_{k+1} \leq 0.1$. This completes the proof. □

Theorem 3.1. *Under the assumptions of Lemma 3.1 S_0 with k-fold iterative refinement is blockwise forward stable. There exists k_0 depending only on n such that for every $k \geq k_0$*

$$\frac{\|x_k - x^*\|_2}{\|x^*\|_2} \leq 2.1L\varepsilon_M\ cond_\mu(A; x^*). \tag{22}$$

Proof. We apply the results of Lemma 3.1. By assumptions (13), we have

$$q_{k+1} \leq q_k(0.1 + 2.5 * 0.01) + 2.5L\varepsilon_M \; cond_\mu(A; x^*),$$

so

$$q_{k+1} \leq q_k 0.2 + 2.5L\varepsilon_M \; cond_\mu(A; x^*).$$

From this it follows that

$$q_{k+1} \leq (0.2)^k + 2L\varepsilon_M \; cond_\mu(A; x^*).$$

From this (22) follows immediately. □

Remark 3.1. Similar results can also be obtained for classical iterative refinement. However, in this case, in (15) we have

$$q_{k+1} = q_k q_0 + 2.5L\varepsilon_M \; (cond_\mu(A; x^*) + q_k \kappa_\mu(A)). \tag{23}$$

Clearly, this sequence $\{q_k\}$ converges more slowly than in the case of k-fold iterative refinement. We also see that S_0 with classical iterative refinement is blockwise forward stable.

4. Numerical experiments

We now give some numerical tests to illustrate our theoretical results of the previous sections. All tests were carried in *MATLAB*, version 6*.5.0.180913a (R13)* with unit roundoff $\varepsilon_M \approx 2.2 \cdot 10^{-16}$ in IEEE double precision.

Let $x^* = A^{-1}b$ be the exact solution to $Ax = b$ and let x_k be the computed approximation to x^* by IR or RIR, respectively.

We report the following statistics for each iteration:

- blockwise relative forward error: $\gamma_{\mu}(A,b,x_k) = \frac{\|x_k - x^*\|_2}{cond_{\mu}(A;x^*)\ \|x^*\|_2}$,

- normwise relative backward error: $\beta_{norm}(A,b,x_k) = \frac{\|b - Ax_k\|_2}{\|A\|_2\ \|x_k\|_2}$,

- blockwise relative backward error: $\beta_{\mu}(A,b,x_k) = \frac{\|b - Ax_k\|_2}{\|\mu(A)\,\mu(x_k)\|_2}$,

- componentwise relative backward error: $\beta_{comp}(A,b,x_k) = \frac{\|b - Ax_k\|_2}{\||A|\,|x_k|\|_2}$.

Example 4.1. We produced the $n \times n$ matrix A and the vector $b(n \times 1)$ with the following *MATLAB* code:

```
A=pascal(n)+1.12e-12*magic(n);
x_star=ones(n,1); %The exact solution is x_star=[1;1;...;1]
b=A*x_star;
```

The command `ones(m,n)` produces an $m \times n$ matrix of ones, and the command `pascal(n)` produces an $n \times n$ matrix from Pascal's triangle, and `magic(n)` is an $n \times n$ matrix constructed from the integers 1 through n^2 with equal row, column, and diagonal sums.

The solver $x_0 = S_0(b)$ computes the approximation x_0 to the exact solution x^* of the system $Ax = b$ with the following *MATLAB* code:

```
x0=A\b; % Gaussian Elimination with Partial Pivoting
x0=x0+1.1e-3*norm(x)*ones(n,1);
```

We partition $A(n \times n)$ as follows

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

where $A_{11}(m \times m)$, with $1 \leq m \leq n$.

The matrix A^{-1} in $\Omega = \mu(A^{-1}\, \mu(A)$ was computed by the *MATLAB* command `inv`.

The results are listed below.

Table 1: Results for the computed solutions to $Ax = b$ for $n = 10$, $m = 5$, and Iterative Refinement (IR). Here $\kappa_2(A) = 4.1552 \cdot 10^9$, $\kappa_{mu} = 4.6485 \cdot 10^8$, and $cond_\mu(A, b, x^*) = 2.7331 \cdot 10^8$.

k	$\gamma_\mu(A, b, x_k)$	$\beta_{norm}(A, b, x_k)$	$\beta_\mu(A, b, x_k)$	$\beta_{comp}(A, b, x_k)$
0	$1.2683e-011$	$1.8354e-003$	$2.5556e-003$	$3.4664e-003$
1	$4.4272e-014$	$6.4066e-006$	$8.9205e-006$	$3.4664e-003$
2	$1.5403e-016$	$2.2286e-008$	$3.1030e-008$	$4.2090e-008$
3	$2.4762e-018$	$7.7567e-011$	$1.0800e-010$	$1.4650e-010$
4	$5.6001e-017$	$2.9400e-011$	$4.0936e-011$	$5.5526e-011$
5	$1.6620e-018$	$2.8972e-011$	$4.0340e-011$	$5.4718e-011$
6	$3.5112e-017$	$1.8458e-011$	$2.5700e-011$	$3.4860e-011$
7	$3.5663e-017$	$4.3422e-013$	$6.0460e-013$	$8.2009e-013$
8	$6.3066e-017$	$1.3808e-011$	$1.9226e-011$	$2.6078e-011$
9	$3.0336e-017$	$1.6486e-011$	$2.2955e-011$	$3.1136e-011$
10	$2.1622e-017$	$4.3869e-012$	$6.1082e-012$	$8.2852e-012$
100	$7.6738e-017$	$1.1605e-011$	$1.6159e-011$	$2.1918e-011$
1000	$2.6456e-017$	$9.3592e-012$	$1.3032e-011$	$1.7676e-011$

These numerical results indicate that k-fold iterative refinement is very

Table 2: Results for the computed solutions to $Ax = b$ for $n = 10$, $m = 5$, and Recurrent Iterative Refinement (RIR). Here $\kappa_2(A) = 4.1552 \cdot 10^9$, $\kappa_{mu} = 4.6485 \cdot 10^8$, and $cond_\mu(A, b, x^*) = 2.7331 \cdot 10^8$.

k	$\gamma_\mu(A, b, x_k)$	$\beta_{norm}(A, b, x_k)$	$\beta_\mu(A, b, x_k)$	$\beta_{comp}(A, b, x_k)$
0	$1.2683e-011$	$1.8354e-003$	$2.5556e-003$	$3.4664e-003$
1	$4.4272e-014$	$6.4066e-006$	$8.9205e-006$	$1.2100e-005$
2	$2.9287e-018$	$7.7521e-011$	$1.0794e-010$	$1.4641e-010$
3	$5.0335e-017$	$3.9907e-017$	$5.5566e-017$	$7.5371e-017$
4	$4.3737e-018$	$1.7882e-017$	$2.4899e-017$	$3.3773e-017$

stable and robust. Iterative refinement also provides an effective way to make almost every solver S_0 forward stable but not backward stable. We suggest to use a few S_k ($k = 1, \ldots, 4$ instead of a few steps of IR, to correct results.

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

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

[4] 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.

[5] A. Buttari, J. Dongarra, J. Langou, J. Langou, J.P. Luszczek, J. Kurzak, Mixed Precision Iterative Refinement Techniques for the Solution of Dense Linear Systems, International Journal of High Performance Computing Applications 21(4) (2007) 457–466.

[6] J. Demmel, Y. Hida, W. Kahan, X.S. Li, S. Mukherjee, E.J. Riedy, Error bounds from extra-precise iterative refinement, ACM TOMS, Vol. 32 (2) (2006) 325–351.

[7] E. Deutsch, Matricial norms, Numer.Math. 16 (1970) 73–84.

[8] J. Głuchowska, A. Smoktunowicz, Solving the linear least squares problem with very high relative accuracy, Computing 45 (1990) 345–354.

[9] M. Jankowski, H. Woźniakowski, Iterative refinement implies numerical stability, BIT 17 (1977) 303–311.

[10] A. Kiełbasiński, Iterative refinement for linear systems in variable-precision arithmetics, BIT 21 (1981) 97–103.

[11] A.M. Ostrowski, On some metrical properties of operator matrices and matrices partitioned into blocks, J.Math.Anal.Appl. 2 (1961) 161–209.

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

[13] A. Smoktunowicz, Blockwise analysis for solving linear systems of equations, J. KSIAM Vol. 3 No. 1 (1999) 31–41.

[14] A. Smoktunowicz, A note on the strong componentwise stability of algorithms for solving symmetric linear systems, Demonstratio Math. Vol. XXVIII No 2 (1995) 443–448.

[15] A. Smoktunowicz, Block matrices and symmetric perturbations, Linear Algebra Appl. 429 (2008) 2628–2635.

[16] A. Smoktunowicz, J. Sokolnicka, Binary cascades iterative refinement in doubled-mantissa arithmetic, BIT 24 (1984) 123–127.

[17] J.L. Rigal, J. Gaches, On the comparability of a given solution with the data of a linear system, J.Assoc. Comput.Mach. 14(3) (1967) 543–548.