Performance Evaluation of Multiple and Mixed Precision Iterative Refinement Method and its Application to High-Order Implicit Runge-Kutta Method

Tomonori Kouya
Shizuoka Institute of Science and Technology, Shizuoka 437-8555, Japan
tkouya@cs.sist.ac.jp

Abstract

Buttari et al. have proposed mixed precision iterative refinement method using the IEEE754 single and double precision arithmetic for solving linear systems of equations. We broaden the scope of the applications of the mixed precision iterative refinement method by using a combination of double precision arithmetic and multiple precision arithmetic and show that the new method has higher performance and yields the same level precise solutions as the original method. Finally, through our numerical experiments, we demonstrate that the fully implicit Runge-Kutta methods with the mixed precision iterative refinement can speedup.
Keywords: linear system, iterative refinement, multiple precision floating-point arithmetic, implicit Runge-Kutta method

1 Introduction

Nowadays, it is necessary for large-scale scientific computations to guarantee the user-required accuracy of their numerical results. In large-scale computation, the chances of ill conditions occurring are high. Hence, round-off errors may accumulate in their numerical processes, which use standard IEEE single precision (SP) or double precision (DP) arithmetic, and hence, their accuracy is affected. In order to prevent such a situation, we sometimes need to use multiple precision (MP) floating-point arithmetic such as quadruple, octuple, and hextuple precision in large-scale scientific computation.

The MP floating-point arithmetic in use generally is implemented as a software library, and hence, it is much slower than SP, DP, or integer arithmetic directly running on hardware units in CPUs. Therefore, it is desirable for users to select the least digits of MP floating-point arithmetic that can guarantee them the numerical accuracy they require.

The SP-DP type mixed precision iterative refinement method [2] proposed by Buttari et al. in 2007, is an economical and stable method to guarantee an accurate solution of linear equations when its illness is not so large in SP

floating-point arithmetic. In this method, the numerical residual is obtained in high-cost DP floating-point arithmetic, and then, a more complex direct method with the DP residual as a constant vector is executed in low-cost SP floating-point arithmetic. Through these processes, we can save computational time and can obtain a numerical result that is as accurate as the result obtained we use simply DP direct method. This idea can be easily expanded to MP floating-point computation. As a result, the MP scientific computation including solvers of linear equations by using the mixed precision iterative refinement method can be economized because running cost of the software MP floating-point arithmetic mostly depends on its digits. This method is particularly desirable for the inner process in high order implicit Runge-Kutta (IRK) method. The linear equations appearing in the IRK method are relatively well-conditioned, so we hope to minimize the computational costs of the solving processes by using a mixed-precision iterative refinement method.

In this paper, we first explain the mixed precision iterative refinement method proposed by Butttari et al. and confirm that it can be applied in a multiple precision environment. Second, we will experiment MP-MP (multiple precisions) and DP-MP type iterative refinement methods applied to well- and ill-conditioned problems in a standard PC environment with a multi-core CPU and demonstrate that our proposed DP-MP type iterative refinement method can obtain the maximum speedup ratio. In addition, we show the limitations in speedup by parallelizing these iterative refinement methods in a multi-core CPU. Finally, we benchmark the fully implicit Runge-Kutta methods with the DP-MP iterative refinement method that performed the best to show its advantage.

2 Theory and Algorithm of Mixed Precision Iterative Refinement Method

In 1967, C. Moler proposed the original iterative refinement method for solving linear systems of equations. His idea was based on the Newton method for application to a n-th dimensional equation.

We suppose that the targeted linear system of equations is

$$\begin{aligned} &A\mathbf{x} = \mathbf{b} \\ &A \in \mathbb{R}^{n\times n},\ \mathbf{x} \in \mathbb{R}^n,\ \mathbf{b} \in \mathbb{R}^n, \end{aligned} \tag{1}$$

where the coefficient matrix A is always normal. In this paper, any elements of A and $\mathbf{b}$ in (1) are given in any expected precision.

In this case, the algorithm of the iterative refinement method for solving the linear systems of equations (1) is as follows:

$$\mathbf{r}_k := \mathbf{b} - A\mathbf{x}_k \tag{2}$$

$$\text{Solve } A\mathbf{z}_k = \mathbf{r}_k \text{ for } \mathbf{z}_k \tag{3}$$

$$\mathbf{x}_{k+1} := \mathbf{x}_k + \mathbf{z}_k \tag{4}$$

Buttari et. al proved that the mixed precision iterative refinement method can obtain an approximation with the same level of relative errors as obtained by using standard methods for solving linear systems of equations, which purely uses L decimal digits floating-point arithmetic. Moreover, their proposed mixed

precision iterative refinement method can gain better performance if (3) is calculated with $S(< L)$ decimal digits. They also maintain that the algorithm employed in (3) must be numerically stable, concretely like the GMRES method or direct method. We employ the direct method by LU factorization with partial pivoting as a solver for (3). In this case, (3) is expressed as

$$(PLU)\mathbf{z}_k = \mathbf{r}_k.$$

Before the iterations, A must be factorized as PLU, where P is a permutation matrix obtained by partial pivoting. The forward and backward substitutions are only executed during the iteration process. The algorithm corresponding to the formulas (2) - (4) is as follows:

1. $A^{[L]} := A$, $A^{[S]} := A^{[L]}$, $\mathbf{b}^{[L]} := \mathbf{b}$, $\mathbf{b}^{[S]} := \mathbf{b}^{[L]}$
2. $A^{[S]} := P^{[S]}L^{[S]}U^{[S]}$
3. Solve $(P^{[S]}L^{[S]}U^{[S]})\mathbf{x}_0^{[S]} = \mathbf{b}^{[S]}$ for $\mathbf{x}_0^{[S]}$
4. $\mathbf{x}_0^{[L]} := \mathbf{x}_0^{[S]}$
5. For $k = 0, 1, 2, ...$
   (a) $\mathbf{r}_k^{[L]} := \mathbf{b}^{[L]} - A\mathbf{x}_k^{[L]}$
   (b) $\mathbf{r}_k^{[S]} := \mathbf{r}_k^{[L]}$
   (c) Solve $(P^{[S]}L^{[S]}U^{[S]})\mathbf{z}_k^{[S]} = \mathbf{r}_k^{[S]}$ for $\mathbf{z}_k^{[S]}$
   (d) $\mathbf{z}_k^{[L]} := \mathbf{z}_k^{[S]}$
   (e) $\mathbf{x}_{k+1}^{[L]} := \mathbf{x}_k^{[L]} + \mathbf{z}_k^{[L]}$
   (f) Exit if $\|\mathbf{r}_k^{[L]}\|_2 \leq \sqrt{n}\ \varepsilon_R\ \|A\|_F\|\mathbf{x}_k^{[L]}\|_2 + \varepsilon_A$,

where $A^{[S]}$ or $\mathbf{b}^{[L]}$ denotes the approximated matrix or vector, respectively, rounded to S or L decimal digits floating-point numbers.

Below, we will describe the conditions for the convergence of the S- L decimal digits mixed precision iterative refinement method.

The symbols ε_S and ε_L denote the machine epsilons in S and L decimal digits floating-point arithmetic, respectively. In L digits arithmetic, (2) can be expressed as

$$\begin{aligned} &\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k + \mathbf{e}_k, \\ &\quad \text{where} \ \ \|\mathbf{e}_k\| \leq \varphi_1(n)\varepsilon_L \left(\|A\| \cdot \|\mathbf{x}_k\| + \|\mathbf{b}\|\right). \end{aligned} \tag{5}$$

The residual $\mathbf{r}_k$ includes the computational error $\mathbf{e}_k$[5]. Similarly, we can express (4) as

$$\begin{aligned} &\mathbf{x}_k = \mathbf{x}_k + \mathbf{z}_k + \mathbf{f}_k, \\ &\quad \text{where } \|\mathbf{f}_k\| \leq \varphi_2(n)\varepsilon_L \left(\|\mathbf{x}_k\| + \|\mathbf{z}_k\|\right). \end{aligned} \tag{6}$$

Moreover, (3) can also be expressed as

$$\begin{aligned} &(A + H_k)\mathbf{z}_k = \mathbf{r}_k, \\ &\quad \text{where } \|H_k\| \leq \phi(n)\varepsilon_S\|A\|. \end{aligned} \tag{7}$$

At this time, we define α_F and $\beta_F \in \mathbb{R}$ as follows:

$$\begin{aligned}\alpha_F &= \frac{\phi(n)\kappa(A)\varepsilon_S}{1-\phi(n)\kappa(A)\varepsilon_S} + 2\varphi_1(n)\kappa(A)\varepsilon_L + \varphi_2(n)\varepsilon_L \\ &\quad + 2(\varphi_1(n)\varepsilon_L)\varphi_2(n)\kappa(A)\varepsilon_L \\ &= \psi_F(n)\kappa(A)\varepsilon_S \qquad (8) \\ \beta_F &= 4\varphi_1(n)\kappa(A)\varepsilon_L + \varphi_2(n)\varepsilon_L + 4(1+\varphi_1(n)\varepsilon_L)\varphi_2(n)\kappa(A)\varepsilon_L \\ &= \rho_F(n)\kappa(A)\varepsilon_L \qquad (9)\end{aligned}$$

If the conditions

$$\frac{\rho_F(n)\kappa(A)\varepsilon_S}{1-\psi_F(n)\kappa(A)\varepsilon_S} < 1 \quad \text{and} \quad \alpha_F < 1 \qquad (10)$$

are satisfied, we can expect that

$$\lim_{k\to\infty} \|\mathbf{x}-\mathbf{x}_k\| \leq \frac{\beta_F}{1-\alpha_F}\|\mathbf{x}\|. \qquad (11)$$

This implies that the normwise relative error in the approximation $\mathbf{x}_k$ can reduce the order of $\beta_F/(1-\alpha_F)$ [2].

For the above mentioned conditions, the S-L digits mixed precision iterative refinement can converge if

$$\kappa(A)\varepsilon_S << 1 \qquad (12)$$

must be satisfied. From the above condition, for the convergence of the S-L decimal digits mixed precision iterative refinement method, it is necessary that S is a large computational digit if $\kappa(A)$ is large. In this case, it requires more computational cost, and consequently its effectiveness would decrease. On the other hand, the question why we require $L(> S)$ digits approximation would arise when $\kappa(A)$ is small.

Therefore, the cases when the S-L decimal digits mixed precision iterative refinement method will be advantageous are as follows:

- to require over L digits approximation if $\varepsilon_S^{-1} > \kappa(A)$
- to be in computational environment that S digits arithmetic can be executed much faster than L digits arithmetic

and vice versa. Figure 1 explains such conditions.

3 Strategy for Selecting Precision used in Iterative Refinement Method

The computational costs of multiple precision floating-point arithmetic are more than that of the quadruple precision implemented arithmetic because the software libraries generally require much more computational cost than IEEE754 single or double precision floating-point arithmetic embedded as hardware units on CPUs. For this reason, the mixed precision iterative refinement method which uses S decimal digits and L decimal digits floating-point arithmetic, can achieve a better performance than the direct methods, which purely uses L decimal digits floating-point arithmetic, if users require U decimal digits approximation more than the quadruple precision, and U is much less than $\log_{10}\kappa(A)$,

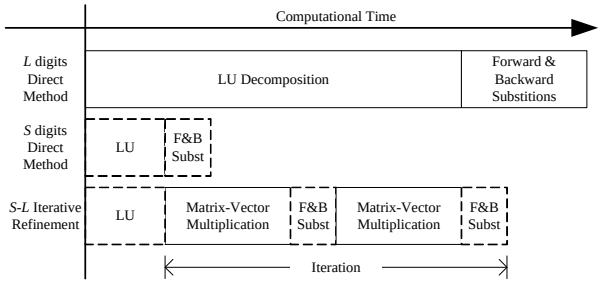

Figure 1: Structure of computational time of mixed precision iterative refinement

where $L > U + \log_{10} \kappa(A)$ and $S > \log_{10} \kappa(A)$. The mixed precision iterative refinement method also has approximations at the same level of relative errors as in direct methods. Hence, we select 3 combinations of precision (SP-DP, DP-MP and MP-MP) to be applied in mixed precision iterative refinements, which can optimize the level of relative errors and computational costs.

1. $\kappa(A) < 10^7 \implies$ Single Precision($S = 7$)-Double Precision($L = 15$): SP-DP type
2. $\kappa(A) < 10^{15} \implies$ Double Precision($S = 15$)-Quadruple Precision or Multiple Precision($L > 30$): DP-MP type
3. $\kappa(A) > 10^{15} \implies$ Quadruple Precision or Multiple Precision - Multiple Precision : MP-MP type

The combination of DP-MP and MP-MP iterative refinement methods in above combinations can achieve a better performance than the original SP-DP iterative refinement method proposed by Buttari et al. We can expect the DP-MP iterative refinement method to achieve the best performance among them because it uses high-speed hardware computation (DP) and slow software computation (MP).

However, the actual performances of these mixed precision iterative refinements depend on the computation environments on which they are executed. As shown in Figure 1, the S-L decimal digits iterative refinement method would be meaningless if the S decimal digits computation could not be executed faster than the L decimal digits one. For this reason, we set S and L as $L/S \geq 2$ in our numerical experiments described in this paper. Minimize L/S ratio to an extent where better performance is achieved with the mixed precision iterative refinement is the objective of future works.

4 Performance Evaluation

In this section, we evaluate the performances of the MP-MP and DP-MP iterative refinement methods in the following environment:

CPU AMD Phenom II X6 1065T (6 cores)

RAM 16 GB

OS Scientific Linux 6 x86_64

C compiler gcc 4.4.5

Libraries BNCpack[6] 0.7 with MPFR 3.1.0[7]/GMP 5.0.2[1] and Pthread

LAPACK LAPACK 3.4.0 and ATLAS[8] 3.8.3

The DP computations are executed by using BNCpack without using the advantages of the CPU architectures, original LAPACK (compiled with gfortran), and its tuned ATLAS.

The MP computations are executed by using BNCpack based on MPFR/GMP. Multiple precision floating-point variables provided by MPFR and GMP are able to have any length of bits of mantissas, so mixed precision computations are freely executed in any positions of codes.

We set the convergence check such as

$$\varepsilon_R := \varepsilon_L,\ \varepsilon_A := 0, \tag{13}$$

to obtain the best approximation so long as we use less than L decimal digits in the computation, where ε_L is the machine epsilon in the L decimal digits computation.

4.1 Performance Evaluation of MP-MP Iterative Refinement Method

We evaluate the performance of the MP-MP iterative refinement method by using the Lotkin matrix (14) as an example of ill- conditioned problems.

$$A = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1/2 & 1/3 & \cdots & 1/(n+1) \\ \vdots & \vdots & & \vdots \\ 1/n & 1/(n+1) & \cdots & 1/(2n-1) \end{bmatrix} \tag{14}$$

We employ a true solution as $\mathbf{x} = [1\ 2\ ...\ n]^T$ and create test problems with correctly rounded L decimal digits A and $\mathbf{b} = A\mathbf{x}$.

Various dimensions $n = 128, 256, 512$ and 1024 used in our numerical experiment, the corresponding condition number $\kappa_\infty(A)(= \|A\|_\infty \|A\|_\infty)$ of the Lotkin matrices, and the S and L decimal digits for which we select the precision for the MP floating- point arithmetic are shown in Table 1.

Table 1: Condition number of Lotkin matrix A and sets of selected decimal digits

Dimension	128	256	512	1024
$\log_{10}(\kappa_\infty(A))$	195	391	784	1576
S (decimal)	250	500	1000	1750
L (decimal)	500	1000	2000	3500

Table 2: Computational time(s) and significant decimal digits of direct method(L and S), MP-MP mixed precision iterative refinement method (iterative times in parenthesis)

(n, L)	$(128, 500)$		$(256, 1000)$		$(512, 2000)$		$(1024, 3500)$	
	s	digits	s	digits	s	digits	s	digits
$*$	0.2	67	6.7	362	79	222	1449	187
$**$	0.3(6)	308	8(4)	1112	91(6)	1219	1637(11)	1936
$***$	0.5	309	17	1114	259	1220	3988	1936

$*$ ... Direct Method (S)

$**$ ... Iterative Refinement Method ($S - L$)

$***$ ... Direct Method (L)

As mentioned previously, the condition $S >> \kappa(A)$ must be satisfied if the mixed precision iterations have to be converge properly. These selected precisions afford to converge our numerical examples.

We present the results of the performance evaluation in Table 2.

As we had previously shown the significant decimal digits of the numerical solution are of the same order as that obtained using the L digits direct method, and the computational time added the cost of the residual computation to the cost of the S digits direct method.

In order to speed up the MP-MP type iterative refinement method, we parallelize it by using a parallelized direct method and a parallelized residual computation with Pthread on a multi-core CPU. The efficiency of the parallelized MP-MP iterative refinement methods is shown in Figure 2.

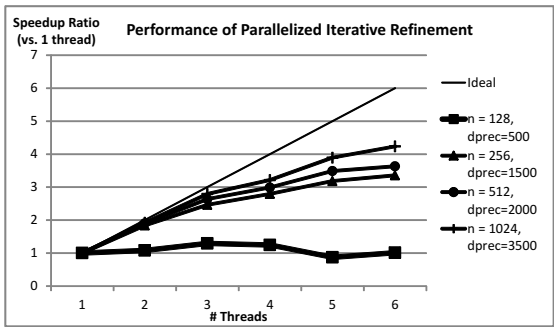

Figure 2: Performance of parallelized MP-MP iterative refinement method

In cases of $n = 128$ and 500 decimal digits, our parallelization is not completely efficient. If anything, the parallelization limits the efficiency. We can recognize that large-scale problems are more efficient, i.e. the DP-MP type iterative refinement method is not completely efficient because of such parallelization limits.

4.2 Performance Evaluation of DP-MP Iterative Refinement Method

Here we evaluate the DP-MP iterative refinement method by using well-conditioned linear system of equations (15).

We create the dense square matrix with a fixed condition number by multiplying the normal matrix X generated by using a standard random generator, its inverse matrix X^{-1}, and the diagonal matrix $D = \mathrm{diag}(n, n-1, ..., 1)$ such as

$$A = XDX^{-1}. \tag{15}$$

Thus, we can obtain the well-conditioned matrix A with $\kappa_2(A) = n$. We employ a true solution as $\mathbf{x} = [1\ 2\ ...\ n]^T$ and create test problems with correctly rounded L decimal digits A and $\mathbf{b} = A\mathbf{x}$.

In the process of the DP-MP type iterative refinement method, the LU factorization outside iterations and forward and backward substitutions inside them are executed in DP arithmetic. On the other hand, the residuals and renewal of approximations inside them are done in MP arithmetic. A combination of these arithmetic can cause a limitation in accuracy for approximation owing to occurring underflow in DP arithmetic. If the length of the exponent in the MP floating-point number is longer than the DP floating-point number, the renewal of approximation cannot be invalid when the norm of the residual is under about 1.0×10^{-308}. For this reason, we limit the computational digits to $L = 50, 100$, and 200 and the dimension of test problems to $n = 128$ to 1024. The results are described as below.

Table 3 shows the relative errors and iterative times of the DP-MP iterative refinement method. For comparison, the results of the MP-MP iterative refinements are shown in this table as well.

Table 3: $\log_{10}$(Relative Error) and iterative times (in parenthesis) of DP-MP iterative refinement

	$L = 50$			
n	MP-MP	BNCpack	LAPACK	ATLAS
128	-47.63 (2)	-49.10 (4)	-49.02 (4)	-49.23 (4)
256	-46.71 (2)	-48.84 (4)	-48.80 (4)	-48.74 (4)
512	-47.24 (2)	-48.09 (4)	-48.41 (4)	-48.76 (4)
1024	-46.96 (2)	-48.75 (4)	-48.61 (4)	-48.32 (4)
n	$L = 100$			
128	-97.38 (2)	-98.94 (7)	-98.69 (7)	-98.93 (7)
256	-96.93 (2)	-99.04 (7)	-98.96 (7)	-99.04 (7)
512	-96.18 (2)	-98.00 (7)	-98.43 (7)	-98.62 (7)
1024	-95.56 (2)	-98.66 (7)	-98.71 (7)	-98.60 (7)
n	$L = 200$			
128	-197.39 (2)	-198.50 (14)	-198.59 (14)	-196.97 (13)
256	-196.38 (2)	-198.65 (14)	-198.68 (14)	-198.71 (14)
512	-196.13 (2)	-198.20 (14)	-198.04 (14)	-198.42 (14)
1024	-196.11 (2)	-198.46 (14)	-198.52 (14)	-198.56 (14)

There is only a slight difference in the relative errors of approximation of the

DP-MP iterative refinement method and the MP-MP type one. In many cases, the DP-MP type can obtain a slightly more precise error of approximation than MP-MP type. When the ratio L/S is larger, the speed of convergence of the iterative refinement method is slower. For this reason, the DP-MP type requires more than 2 to 7 times iterations than the MP-MP type. Figure 3 shows the speedup ratio of the DP-MP type versus that of the MP-MP type.

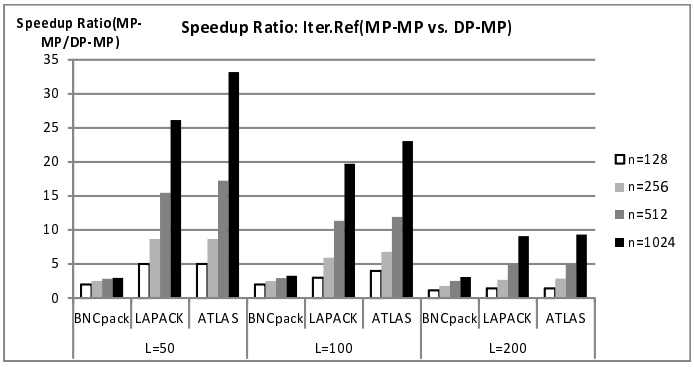

Figure 3: Speedup ratio: MP-MP / DP-MP

In all, we can recognize that more computational digits tend to reduce the speedup ratio. This is because the MP computations of residuals in cases of more iterations reduce the speedup gained by LU factorization, as shown in Figure 3. Despite these drawbacks, the actual DP-MP iterative refinement can perform 33 (in 50 decimal digits) to 10 (in 200 decimal digits) times better with ATLAS and 1.8 to 3.1 times better with BNCpack.

5 Application to Fully Implicit Runge-Kutta Methods with DP-MP Iterative Refinement

As mentioned above, the DP-MP type mixed precision iterative refinement can perform better in relatively well-conditioned linear systems of equations from which the DP direct method can obtain precise solutions. This advantage will be maximized for fully implicit Runge-Kutta (IRK) methods.

For the initial value problem for the n-th dimensional ordinary differential equation (ODE)

$$\left\{ \begin{array}{rcl} \frac{d\mathbf{y}}{dx} & = & \mathbf{f}(x, \mathbf{y}) \\ \mathbf{y}(x_0) & = & \mathbf{y}_0 \end{array} \right. , \tag{16}$$

we discretize it over the integration interval $[x_0, \alpha]$ with constant stepsize $h = (\alpha - x_0)/(2 \cdot 4^l)$. When we obtain a new approximation $\mathbf{y}_{i+1} \approx \mathbf{y}(x_0 + ih)$ from the former one $\mathbf{y}_i$, we must solve the nonlinear system of equations

$$\left\{ \begin{array}{rcl} \mathbf{k}_1 & = & \mathbf{f}(x_i + c_1 h, \mathbf{y}_i + h \sum_{j=1}^{m} a_{1j} \mathbf{k}_j) \\ \mathbf{k}_2 & = & \mathbf{f}(x_i + c_2 h, \mathbf{y}_i + h \sum_{j=1}^{m} a_{2j} \mathbf{k}_j) \\ & \vdots & \\ \mathbf{k}_m & = & \mathbf{f}(x_i + c_m h, \mathbf{y}_i + h \sum_{j=1}^{m} a_{mj} \mathbf{k}_j) \end{array} \right. ,$$

and then, we calculate $\mathbf{y}_{i+1}$ as follows:

$$\mathbf{y}_{i+1} := \mathbf{y}_i + h\sum_{j=1}^{m} w_j \mathbf{k}_j,$$

where m denotes the number of stages in the IRK method, and c_p, a_{pq}, and w_q are constants chosen in the IRK method. In our numerical experiments, we select m stages $2m$-th order ($m = 3, 4, ..., 10$) Gauss type formula[4].

To solve the nonlinear system of equations above, Newton's iteration is often applied as follows:

$$\begin{bmatrix} \mathbf{k}_1^{(l+1)} \\ \mathbf{k}_2^{(l+1)} \\ \vdots \\ \mathbf{k}_m^{(l+1)} \end{bmatrix} := \begin{bmatrix} \mathbf{k}_1^{(l)} \\ \mathbf{k}_2^{(l)} \\ \vdots \\ \mathbf{k}_m^{(l)} \end{bmatrix} - J^{-1}(\mathbf{k}_1^{(l)}, ..., \mathbf{k}_m^{(l)}) \begin{bmatrix} \mathbf{k}_1^{(l)} - \mathbf{f}(x_i + c_1 h, \mathbf{y}_i + h\sum_{j=1}^{m} a_{1j}\mathbf{k}_j^{(l)}) \\ \mathbf{k}_2^{(l)} - \mathbf{f}(x_i + c_2 h, \mathbf{y}_i + h\sum_{j=1}^{m} a_{2j}\mathbf{k}_j^{(l)}) \\ \vdots \\ \mathbf{k}_m^{(l)} - \mathbf{f}(x_i + c_m h, \mathbf{y}_i + h\sum_{j=1}^{m} a_{mj}\mathbf{k}_j^{(l)}) \end{bmatrix},$$

where $J(\mathbf{k}_1^{(l)}, \mathbf{k}_2^{(l)}, ..., \mathbf{k}_m^{(l)})$ denotes

$$J(\mathbf{k}_1^{(l}, \mathbf{k}_2^{(l)}, ..., \mathbf{k}_m^{(l)}) = \left[\begin{array}{c|c|c|c} I_n - J_{11} & -J_{12} & \cdots & -J_{1m} \\ \hline -J_{21} & I_n - J_{22} & \cdots & -J_{2m} \\ \hline \vdots & \vdots & & \vdots \\ \hline -J_{m1} & -J_{m2} & \cdots & I_n - J_{mm} \end{array}\right]. \tag{17}$$

In the above formulas, I_n denotes the n-th dimensional identity matrix and J_{pq} denotes

$$J_{pq} = h a_{pq} \frac{\partial}{\partial \mathbf{y}} \mathbf{f}(x_i + c_p h, \mathbf{y}_i + h\sum_{j=1}^{m} a_{pj}\mathbf{k}_j^{(l)}) \in \mathbb{R}^{n\times n},$$

where $\partial\mathbf{f}/\partial\mathbf{y}$ is the Jacobian matrix of $\mathbf{f}$.

We can expect that the linear system of equations emerged from Newton's iteration have the property that $J(\mathbf{k}_1^{(l}, \mathbf{k}_2^{(l)}, ..., \mathbf{k}_m^{(l)}) \to I_{mn}$ when $h \to 0$. Because of this property, the DP-MP iterative refinement may be applicable to them for the IRK method with small h because they would be well-conditioned in many cases. In such situations, the DP-MP iterative refinement may speed up the IRK method for obtaining MP approximation.

To confirm the hypothesis, we prepare the constant linear ODE

$$\mathbf{f}(x, \mathbf{y}) = -A\mathbf{y},\ \mathbf{y}(0) = [1\ ...\ 1]^T, \alpha = 20$$

with the well-conditioned 128th dimensional matrix $A(\|A\|_1 \approx 10^3)$ based on (15), and then apply the 3 stages 6th order Gauss type $L = 50$ decimal digits IRK method to it. Figure 4 shows the history of the relative errors in the approximations versus stepsize h, and the histories of the condition numbers and Frobenius norms of $J(\mathbf{k}_1^{(l)}, \mathbf{k}_2^{(l)}, ..., \mathbf{k}_m^{(l)})$.

For $h^{-1} = 512$, we can recognize that about 24 decimal digits approximations are obtained and that the condition numbers decrease in 3.8×10^3 to 1.7. These facts verify the applicability of the DP-MP iterative refinements to the linear ODE. We show the speedup ratio of the IRK method with the DP-MP iterative refinement versus that with MP-MP iterative refinement in Figure 5.

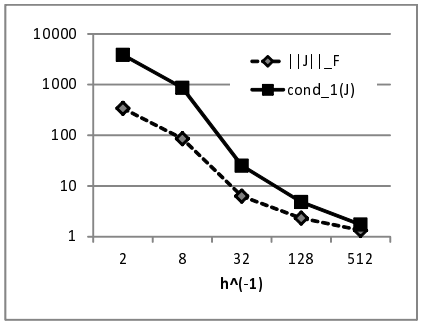

Figure 4: Frobenius norms and condition numbers of J in case of $m = 3$

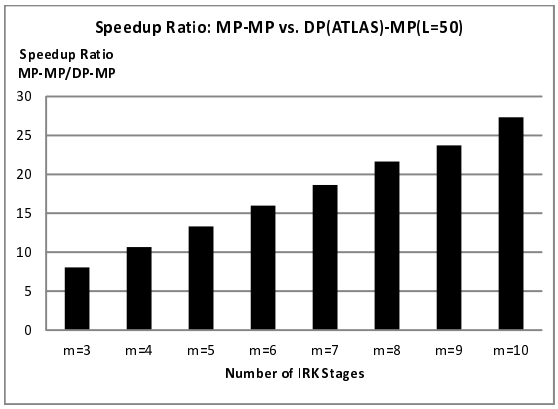

Figure 5: Speedup ratio of implicit Runge-Kutta method (50 Decimal Precision): MP-MP vs. DP(ATLAS)-MP

The IRK method with the DP-MP iterative refinement method can perform about 8 to 27 times (with ATLAS) better than with the MP-MP iterative refinement method.

Finally, we show the results of the accuracy versus computational time with the higher order fully IRK method and the DP(ATLAS)-MP type iterative refinement method in Figure 6. Consequently, the higher order IRK method produce more efficiency and more precise numerical approximation by using the DP-MP iterative refinement method.

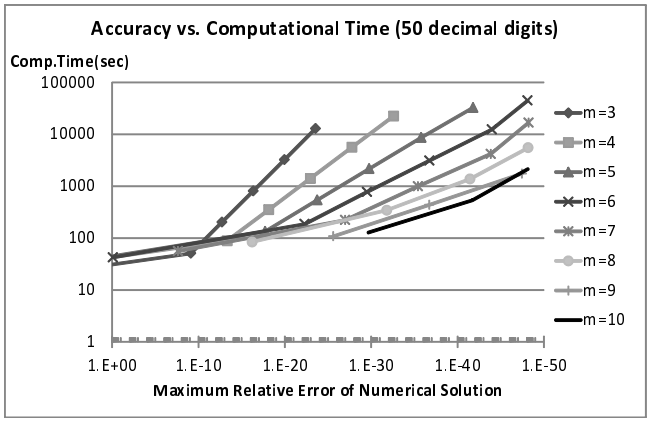

Figure 6: Accuracy vs. computational time of higher order implicit Runge-Kutta method with DP(ATLAS)-MP(50 decimal digits) type iterative refinement method

6 Conclusion and Future Works

The numerical experiments we mentioned above clarify that all types of mixed precision iterative refinements can help us in obtaining approximations at the same level of relative error as the L decimal digits direct method if the sufficient conditions for convergence are satisfied. The experiments also showed that well-tuned LAPACK like ATLAS enabled us to perform the DP-MP iterative refinement method. The application to fully implicit Runge-Kutta methods may provide us the desired efficiency by using the DP-MP type iterative refinement method.

Our future works involve investigation of the mixed precision refinement method's applicability to more general class of initial value problems for ordinary differential equations through various numerical experiments. In particular, the most important problems are time-dependent partial differential equations(PDE). In many cases, the discretization for these PDEs lead to ODEs with sparse matrices, so iterative methods such as the Krylov subspace methods will be more expected than direct methods, as we showed in this paper. Buttari et.al have already proved that an SP-DP type iterative refinement method can expand the performance of the Krylov subspace methods [3]. According to their results, we can expect a better performance for the DP-MP and the MP-MP iterative refinements applied to the Krylov subspace methods.

References

[1] Swox AB. The GNU Multiple Precision arithmetic library. `http://gmplib.org/`.

[2] A.Buttari, J.Dogarra, Julie Langou, Julien Langou, P.Luszczek, and J.Karzak. Mixed precision iterative refinement techniques for the solution of dense linear system. *The International Journal of High Performance Computing Applications*, Vol. 21, No. 4, pp. 457–466, 2007.

[3] A.Buttari, J.Dongarra, J.Kurzakand P.Luszczek, and S.Tomov. Using mixed precision for sparse matrix computations to enhance the performance while achieving 64-bit accuracy. *ACM Trans. Math. Softw.*, Vol. 34, No. 4, pp. 1–22, 2008.

[4] S.P.Nørsett E.Hairer and G.Wanner. *Solving Ordinary Differential Equations*. Springer-Verlag, 1996.

[5] G.W.Stewart. *Matrix Algorithms Volume I: Basic Decompositions*. SIAM, 1998.

[6] Tomonori Kouya. BNCpack. `http://na-inet.jp/na/bnc/`.

[7] MPFR Project. The MPFR library. `http://www.mpfr.org/`.

[8] ATLAS: Automatically Tuned Linear Algebra Software. `http://math-atlas.sourceforge.net/`.

Tomonori Kouya, Ph.D. is interested in multiple precision numerical computation and its application.
Postal Address: Department of Computer Science, Faculty of Comprehensive Informatics, Shizuoka Institute of Science and Technology,
2200-2 Toyosawa, Fukuroi 437-8555 Japan
E-mail: `tkouya@cs.sist.ac.jp`
Web site: `http://na-inet.jp/`