Multiple precision × Guaranteed accuracy + Regularizability × Integrability
Proceedings of Symposium on Celestial Mechanics and N-body Dynamics, 2009
Eds. M. Sekiguchi, M. Shibayama and M. Saito

Mixed Precision iterative refinement for Solving Linear Systems of Equations using IEEE754 Double Precision arithemtic and Multiple Precision arithmetic and its Applicaiton to Fully Implicit Runge-Kutta Method

Tomonori Kouya ⟨tkouya@cs.sist.ac.jp⟩

Shizuoka Institute of Science and Technology

Abstract. Buttari et al. have proposed the mixed precision iterative refinement using the IEEE754 single and double precision arithmetic for solving linear systems of equations. We broaden the scope of applications of the mixed precision iterative refinement by using a combination of double precision arithmetic and multiple precision arithmetic, and show that the new method has higher performance and yields more precise solutions than the original method. Finally, throught our numerical experiments, we demonstrate that the fully implicit Runge-Kutta methods with the mixed precision iterative refinement can speed up.

1. Introduction

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

$$\mathbf{f}(\mathbf{x}) = 0,\ \mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n. \tag{1}$$

In this case, the iteration formula is

$$\mathbf{x}_{k+1} := \mathbf{x}_k - \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_k)\right]^{-1} \mathbf{f}(\mathbf{x}_k), \tag{2}$$

where $\partial\mathbf{f}(\mathbf{x}_k)/\partial\mathbf{x}$ is Jacobian matrix of given function $\mathbf{f}$.

If (1) is the linear system of equations such as

$$\mathbf{f}(\mathbf{x}) = A\mathbf{x} - \mathbf{b},$$

the correspondent Jacobian matrix is given as the constant matrix A. So the Newton iteration changes the following algorithm:

$$\mathbf{r}_k := \mathbf{b} - A\mathbf{x}_k \tag{3}$$

$$\text{Solve } A\mathbf{z}_k = \mathbf{r}_k \text{ for } \mathbf{z}_k \tag{4}$$

$$\mathbf{x}_{k+1} := \mathbf{x}_k + \mathbf{z}_k \tag{5}$$

The above one is the algorithm of iterative refinement for solving linear systems of equations. Althought these iterations are not theoritically necessary, actual finite precision floating-point arithmetic generally create nonzero residual $\mathbf{r}_k$ due to round-off errors occuring in these processes. Therefore several iterations are executed to minimize the norms of residuals in many cases. The residuals must be precise in order to obtain the high precision approximations $\mathbf{x}_k$.

Buttari et al. shows that when the condition number $\kappa(A) = \|A\|\|A^{-1}\|$of A is not so large in comparison with the precision of employed floating-point arithmetic, sufficient condition for convergence of mixed precision iterative refinement is satisfied even if the precision of (4) is lower than the residuals. Their benchmarks clarify that their proposed one in which (4) and (5) are calulated with IEEE754 double precision (simply double precision or DP) and the linear system of equations at 2 with IEEE754 single precision (single precision or SP) can perform better than pure double precision direct method. Their proposed one needs well-tuned computation environment on which single precision computation is certainly faster than double precision computation, such as Cell Broadband Engine or a combination of standard CPUs and well-tuned LAPACK like ATLAS or GotoBLAS.

In this paper, we firstly explain the mixed precision iterative refinement methos proposed by Butttari et al. and confirm that it can be applied with multiplre precision environment. Secondly we will experiment original SP-DP (single precision - double precision), MP-MP (multiple precisions) and DP-MP types iterative refinements applied to well- and ill-conditioned problems on standard PC environment and demonstorate that our proposed DP-MP iterative refinement can obtain the maximum speed-up ratio. Finally, we will benchmark fully implicit Runge-Kutta methods with DP-MP iterative refinement to show its advantage.

2. Theory of Mixed Precision Iterative Refinement

We suppose that the targeted linear system of equations is

$$\begin{aligned} &A\mathbf{x} = \mathbf{b} \\ &A \in \mathbb{R}^{n\times n},\ \mathbf{x} \in \mathbb{R}^n,\ \mathbf{b} \in \mathbb{R}^n, \end{aligned} \tag{6}$$

where the coefficient matrix A is always normal. In this paper, any elements of A and $\mathbf{b}$ in (6) are given in any expected precision.

Buttari et. al prove that mixed precision iterative refinement can obtain the approximation at the same level of relative errors as 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 can gain better performance if (4) is calulated with $S\,(< L)$ decimal digits. They also maintain that the algorithm employed in this part must be numerically stable, concretely like GMRES method or direct method. We employ the direct method by LU factorization with partial pivoting as solver for (4). In this case, (4) is expressed as

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

Before iterations, A must be factorized as PLU, where P is a permutation matrix by partial pivoting. Forward and backward substitutions are only executed in the process of iterations.

The algorithm corresponding to the formulas (3) - (5) 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 $\mathbf{b}^{[L]}$ means the approximated matrix or vector, respectively, rounded to S or L decimal digits floating-point numbers.

We will describe the conditions for convergence of the S-L decimal digits mixed precision iterative refinement below.

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, (3) can be expressed as

$$\begin{aligned} \mathbf{r}_k &= \mathbf{b} - A\mathbf{x}_k + \mathbf{e}_k, \\ &\text{where } \|\mathbf{e}_k\| \leq \varphi_1(n)\varepsilon_L\left(\|A\|\cdot\|\mathbf{x}_k\| + \|\mathbf{b}\|\right). \end{aligned} \tag{7}$$

The residual $\mathbf{r}_k$ includes the computational error $\mathbf{e}_k$. Simirly we can express (5) as

$$\begin{aligned} \mathbf{x}_k &= \mathbf{x}_k + \mathbf{z}_k + \mathbf{f}_k, \\ &\text{where } \|\mathbf{f}_k\| \leq \varphi_2(n)\varepsilon_L\left(\|\mathbf{x}_k\| + \|\mathbf{z}_k\|\right). \end{aligned} \tag{8}$$

Moreover (4) can be also expressed as

$$\begin{aligned} (A + H_k)\mathbf{z}_k &= \mathbf{r}_k, \\ &\text{where } \|H_k\| \leq \phi(n)\varepsilon_S\|A\|. \end{aligned} \tag{9}$$

At this time, we denote that $\alpha_F, \beta_F \in \mathbb{R}$ is as follows:

$$\begin{aligned} \alpha_F &= \frac{\phi(n)\kappa(A)\varepsilon_S}{1 - \phi(n)\kappa(A)\varepsilon_S} + 2\varphi_1(n)\kappa(A)\varepsilon_L + \varphi_2(n)\varepsilon_L \\ &\quad + 2(\varphi_1(n)\varepsilon_L)\varphi_2(n)\kappa(A)\varepsilon_L \\ &= \psi_F(n)\kappa(A)\varepsilon_S \qquad (10) \\ \beta_F &= 4\varphi_1(n)\kappa(A)\varepsilon_L + \varphi_2(n)\varepsilon_L + 4(1 + \varphi_1(n)\varepsilon_L)\varphi_2(n)\kappa(A)\varepsilon_L \\ &= \rho_F(n)\kappa(A)\varepsilon_L \qquad (11) \end{aligned}$$

If the conditions

$$\frac{\rho_F(n)\kappa(A)\varepsilon_S}{1 - \psi_F(n)\kappa(A)\varepsilon_S} < 1 \text{ and } \alpha_F < 1 \tag{12}$$

are satisfied, we can expect that

$$\lim_{k\to\infty} \|\mathbf{x} - \mathbf{x}_k\| \leq \frac{\beta_F}{1 - \alpha_F}\|x\|. \tag{13}$$

It means that the normwise relative error in the approximation $\mathbf{x}_k$ can reduce the order of $\beta_F/(1-\alpha_F)$.

As these conditions mentioned above, S-L digits mixed precision iterative refinement can converge if

$$\kappa(A)\varepsilon_S << 1 \tag{14}$$

must be satisfied. According to the large condition number $\kappa(A)$, it needs to be larger computational digits S enought to converge. On the other hand, is requires more computional cost and its advantage would decrease. In addition, the question why we require $L > S$ digits approximation would occure when $\kappa(A)$ is small. The cases that S-L decimal digits mixed precision iterative refinement will be advantageous are :

- 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 vise versa. Figure 1 explains such circumstances.

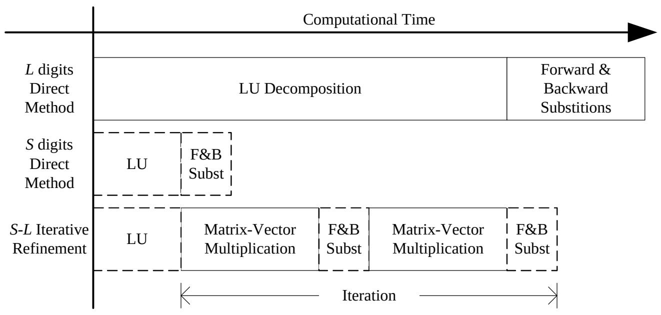

Figure 1. Structure of Computational Time of Mixed-precision iterative refinement

3. Iterative Refinements with Multiple Precision Floating-point Arithmetic

Multple precision floating-point arithmetic more than quadruple precision implemented as software libraries generally needs much more computational cost than IEEE754 single or double precision floating-point arithmetic embedded as hardware units on CPUs. For that reason, mixed precision iterative refinement which uses S decimal digits and L decimal digits floating-point arithmetic, can gain better performance than, and the approximation at the same level of relative errors as by direct methods which purely uses$L$ decimal digits floating-point arithmetic, if users require U decimal digits approximation more than quadruple precision and U is much less than $\log_{10} \kappa(A)$, where $L > U + \log_{10} \kappa(A)$ and $S > \log_{10} \kappa(A)$. So we select 3 combinations of precision (SP-DP, DP-MPand 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 Precison or Mutiple Precision($L > 30$): DP-MP type
3. $\kappa(A) > 10^{15} \implies$ Quadruple Precision or Multiple Precision - Multiple Precision : MP-MP type

DP-MP and MP-MP iterative refinements in above combinations can gain better performance than original SP-DP type one proposed by Buttari et al. We can expect that DP-MP iterative refinement will gain the best performance in them, because it uses high-speed hardware computation (DP) and slow software computation (MP).

But actual performances of these mixed precision iterative refinements depend on the computation enviroments on which they are executed. As Figure 1 maintains, S-L decimal digits iterative refinement would be meaningless if S decimal digits computation could not be executed faster than L decimal digits one. For that reason, we set S and L as $L/S \geq 2$ in our numerical experiments described in this paper. It is one of future works to minimize the ratio L/S enought to gain better performed mixed precision iterative refinement.

4. Performance Evaluation

In this section, we evaluate performances of SP-DP, MP-MP, DP-MP iterative refinements on the following environment:

CPU Intel Core2Quad 6600+

RAM 4GB

OS CentOS 5.2 x86_64

C compiler GCC 4.1.2(gcc, gfortran を使用)

Multiple Precision Library MPFR 2.3.2/GMP 4.2.1 + BNCpack 0.7b

Linear Alegebra Computation Library LAPACK 3.2, ATLAS 3.8.3

SP, DP computations are executed by using BNCpack without taking advantages of CPU architectures, original LAPACK (compiled with gfortran), its tuned ATLAS and GotoBLAS. Core2quad has 4 cores, but we did not use any parallelization.

MP computations are executed by using BNCpack based on MPFR/GMP. Muliple 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

$$\varepsilon_R := \varepsilon_L,\ \varepsilon_A := 0, \tag{15}$$

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

4.1. Performance Evaluation of SP-DP Iterative Refinement

To prepare test problems for numerical experiments, we create the dense square matrix with fixed condition number by multiplying normal matrix X generated by using standard random generator, its inverse matrixX^{-1} and the diagonal matrix $D = \mathrm{diag}(n, n-1, ..., 1)$ such as:

$$A = XDX^{-1}. \tag{16}$$

So we can obtain well- or ill-condition 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 $\mathbf{b} = A\mathbf{x}$. Due to limitation of main memory, the maximum dimension n is 4096. In all cases we create, SP-DP iterative refinement can be converged after 2 or 3 iterations.

First, we show the wall-clock times to execute IEEE754 double precison direct methods (partial pivoting LU decomposion, forward and backward substitutions) and speed-up ratio of single precision direct methods (= wall-clock time of DP direct method/ wall-clock time of SP direct method) in Figure 2.

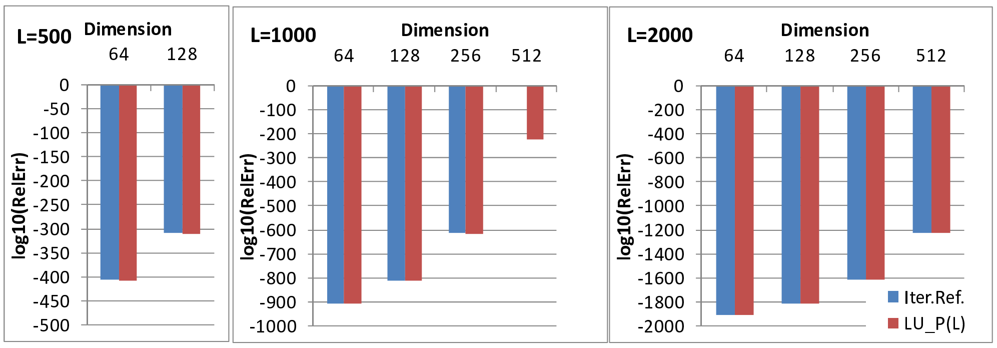

Figure 2. Computational Time (sec) of Double-precision Direct Method(left) and Speedup Ratio of Single-precision Direct Method(right)

In DP computation libraries, the performance of BNCpack is worst and others are at the same level of performance with growing dimensions. GotoBLAS can especially obtain the best performance in small dimensions and the largest speed-up ratio of SP direct method. These benchmarks show that the SP-DP iterative refinement using GotoBLAS can obtain the best performance on our PC environment.

Next, we show speed-up ratios of SP-DP iterative refinement (= wall-clock time of DP direct method / wall-clock time of SP-DP iterative refinement) and maximum elementwise relative errors of approximations obtained by direct and iterative refinements in Figure 3.

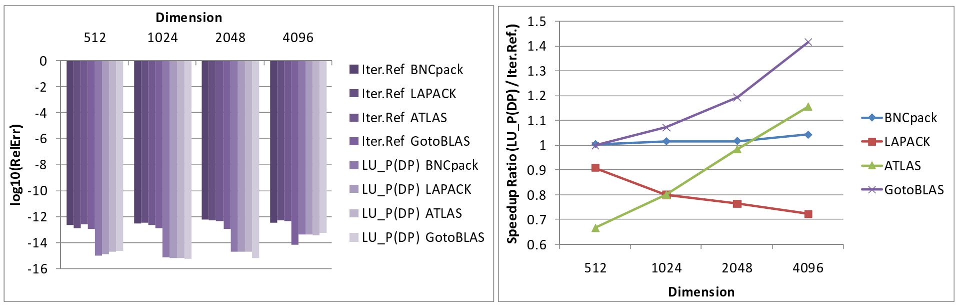

Figure 3. Relative Error of Double-precision Direct Method and SP-DP iterative refinement (left) and Speedup Ratio of SP-DP iterative refinement (right)

We can recognize that relative errors of SP-DP iterative refinements are totally 2 or 3 decimal digits larger than DP direct method, but we cannot maintain that such tendency toward relative error is definite, because the approximation obtained by iterative refinement with GotoBLAS can obtain the best accuracy for 4096 dimentional problem. We also can recognize that the accuracies of approximations obtained by both methods are closer with growing dimensions of problems.

These benchmarks such as Figure 2 attest to well-tuned high performance linear computation libraries able to execute faster SP computation than DP computaion for SP-DP iterative refinement. GotoBLAS produce the largest speed-up ratio as we expected, but LAPACK or ATLAS may be worse.

4.2. Performance Evaluation of MP-MP Iterative Refinement

We evaluate the performance of MP-MP iterative refinement by using the well-conditioned problem (16). Due to limitation of our main memory, we set the dimentions as $n = 128, 256, 512$ and 1024, computaional digits as $L = 50, 100$ and 200. S is fixed as $L/2$. For comparison, the wall-clock times of purely L digits direct methods are shown in Table 1.

Table 1. Computational Time of Multiple-precision Direct Method (sec)

n	$L = 50$	100	200
128	0.15	0.24	0.46
256	1.81	1.97	5.66
512	13.83	23.59	44.7
1024	93.90	160.51	264.94

Figure 4 shows the relative errors and the speedup-ratio of L digits direct methods and MP-MP type ($L/2$-L) iterative refinements.

In similar case of SP-DP type, the approximations obtained by MP-MP iterative refinement can be 2 or 3 decimal digits worse than MP direct method. The speedup ratio can be about 1 to 2 times larger. For the test problems, we can perform better if we select much less S. These results suspect that we can obtain the best performance with DP-MP iterative refinement.

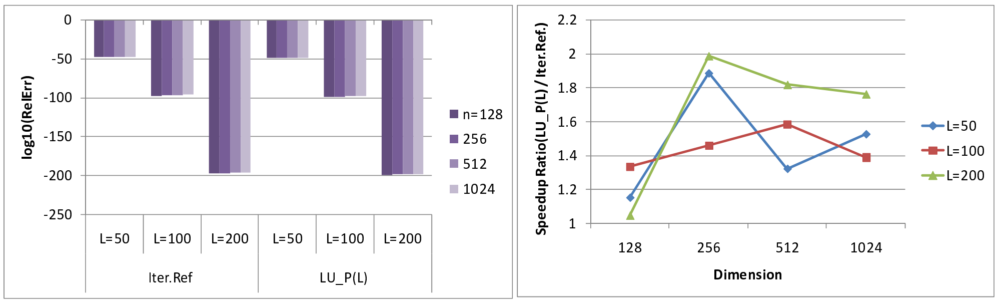

Figure 4. Relative Error of Multiple-precision Direct Method and MP-MP iterative refinement (left) and Speedup Ratio of MP-MP iterative refinement (right)

Next we use Lotkin matrix (17) as a example of ill-conditioned problems.

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1/2 & 1/3 & \cdots & 1/(n+1) \\ \vdots & \vdots & & \vdots \\ 1/n & 1/(n+1) & \cdots & 1/(2n-1) \end{bmatrix} \tag{17}$$

All elements in the first row are replaced to 1 on Hilbert matrix which is well-known as ill-condtionded. The condition number is the same order as Hilbert matrix. If the order of the condition numbers is larger than the number of decimal digits, $A^{[L]}$ rounded in L decimal digits have the smaller condition numbers than true one (Table 2). In this cases, this is the 512th dimensional Lotkin matrix rounded in 500 decimal digits.

Table 2. Condition Numbers of Lotkin Matrices

	$\log_{10}(\kappa_1(A^{[L]}))$		
n	$L = 500$	1000	2000
64	96.0	96.0	96.0
128	193.9	193.9	193.9
256	389.8	389.8	389.8
512	506.0	781.6	781.6

Table 3. Computational Time of Multiple-precision Direct Method (sec): $L = 500, 1000$ and 2000

	Comp.Time (sec)		
n	$L = 500$	1000	2000
64	0.19	0.57	2.19
128	1.51	4.50	17.47
256	12.15	35.96	139.51
512	97.00	286.76	1115.88

For that reasons, MP-MP iterative refinement in 500 decimal digits computaion can converge under 128th dimentional problems, under 256th dimesional ones in 1000 decimal digits computaion, and under 512th dimensional ones in 2000 decimal digits computation. As shown in Figure 5, the relative errors of approximations is at the same level as L decimal digits direct method if converged. In case of 1000 decimal digits and 512th dimensional problem, direct method can obtain 200 decimal digits approximation but MP-MP iterative refinement cannot converge due to lack of accuracy about 800 decimal digits ($S = 500$). If it can converge, more decimal digits can speed up MP-MP iterative refinement though the number of iterations need more additional 3 to 6 iterations. In similar case of for well-conditioned problems, more decimal digits MP-MP iterative refinement can perform about 1.5 to 3.4 times much better MP than direct method as shown in Figure 6).

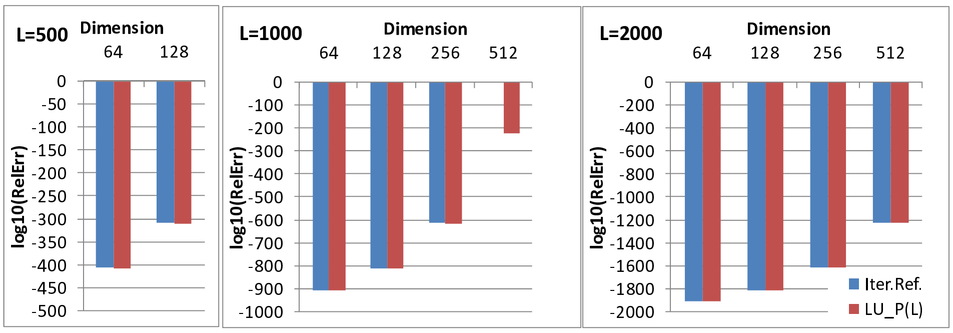

Figure 5. Relative Error of Multiple-precision Direct Method and MP-MP iterative refinement for Lotkin Matrix: $L = 500$(left), $L = 1000$ (Middle) and $L = 200$ (right)

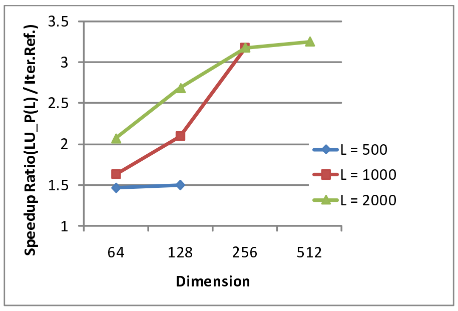

Figure 6. Speedup Ratio of MP-MP iterative refinement: Lotkin Matrix

4.3. Performance Evaluation of DP-MP Iterative Refinement

We here evaluate DP-MP iterative refinement by using well-conditions linear system of equations (16). In this algorithm, 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 cause limitation of accurary for approximation due to occuring underflow in DP arithmetic. If the length of exponent in MP floatint-point number is longer than DP's one, the renewal of approximation cannot be invalid when the norm of residual is under about 1.0×10^{-308}. For that reason, we limit the computational digits to $L = 50, 100$ and 200 and the dimension of test problems $n = 128$ to 1024. The results are described as below.

Table 4 show the relarive erros and iterative times of DP-MP iterative refinement. For comparison, the results of MP-MP iterative refinements are shown in it.

Table 4. $\log_{10}$(Relative Error) and Iterative Times (in Parethesis) of DP-MP iterative refinement

	$L = 50$				
n	MP-MP	BNCpack	LAPACK	ATLAS	GotoBLAS
128	-47.63 (2)	-49.10 (4)	-49.02 (4)	-49.23 (4)	-49.21 (4)
256	-46.71 (2)	-48.84 (4)	-48.80 (4)	-48.74 (4)	-49.12 (4)
512	-47.24 (2)	-48.09 (4)	-48.41 (4)	-48.76 (4)	-48.79 (4)
1024	-46.96 (2)	-48.75 (4)	-48.61 (4)	-48.32 (4)	-48.36 (4)
n	$L = 100$				
128	-97.38 (2)	-98.94 (7)	-98.69 (7)	-98.93 (7)	-98.86 (7)
256	-96.93 (2)	-99.04 (7)	-98.96 (7)	-99.04 (7)	-98.94 (7)
512	-96.18 (2)	-98.00 (7)	-98.43 (7)	-98.62 (7)	-98.60 (7)
1024	-95.56 (2)	-98.66 (7)	-98.71 (7)	-98.60 (7)	-98.64 (7)
n	$L = 200$				
128	-197.39 (2)	-198.50 (14)	-198.59 (14)	-196.97 (13)	-198.64 (13)
256	-196.38 (2)	-198.65 (14)	-198.68 (14)	-198.71 (14)	-198.38 (13)
512	-196.13 (2)	-198.20 (14)	-198.04 (14)	-198.42 (14)	-198.56 (13)
1024	-196.11 (2)	-198.46 (14)	-198.52 (14)	-198.56 (14)	-195.25 (13)

There is very little difference between the relative errors of approximation of DP-MP iterative refinement and MP-MP type one. In many cases, DP-MP type can obtain a little bit more precise than MP-MP type. When the ratio L/S is larger, the speed of convergence of iterative refinement is slower. For that reason, DP-MP type requres more than 2 to 7 times iterations than MP-MP type. Figure 7 shows the speedup ratio versus MP-MP type.

Totally we can recognize that more computational digits tend to reduce the speedup ratio. This is because MP computations of residuals in more iterations reduce the speedup gained by LU factorization as Table 4 shows. Even though there are such disadvantages, actual DP-MP iterative refinement can perform 10 (in 50 decimal digits) to 30 (in 200 decimal digits) times better with GotoBLAS, 1.8 to 4 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 precition iterative refinement can perform better to relatively well-contioned linear systems of equations from which DP direct method can obtain precise solutions. The advantage will be maximazed for fully implicit Runge-Kutta (IRK) methods.

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

$$\left\{ \begin{array}{rcl} \frac{d\mathbf{y}}{dx} & = & \mathbf{f}(x, \mathbf{y}) \\ \mathbf{y}(x_0) & = & \mathbf{y}_0 \end{array} \right. , \tag{18}$$

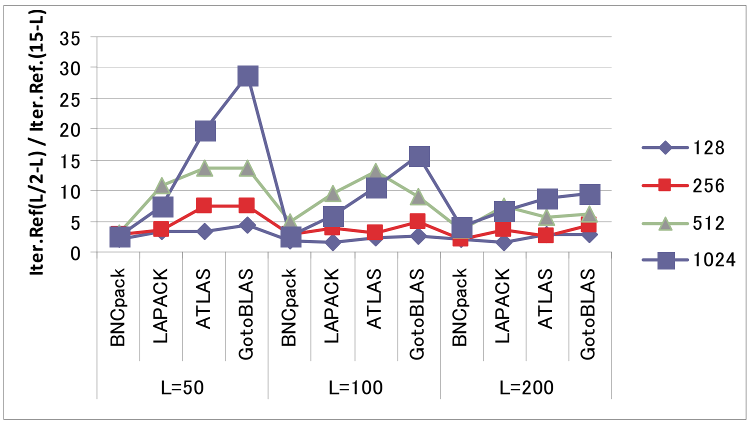

Figure 7.   Speedup ratio: MP-MP / DP-MP

we discretize it over integration interval $[x_0, \alpha]$ with constant stepsize $h = (\alpha - x_0)/(2 \cdot 4^l)$. When we obtain 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

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

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 means a number of stages of IRK method, and c_p, a_{pq} and w_q are contants chosen in IRK method. In our numerical experiments, we select 3 stages 6th order Gauss type formula.

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

$$J(\mathbf{k}_1^{(l}, \mathbf{k}_2^{(l)}, ..., \mathbf{k}_m^{(l)}) = \left[\begin{array}{c|c|c|c} I_n - J_{11} & -J_{12} & \cdots & -J_{1m} \\ \hline -J_{21} & I_n - J_{22} & \cdots & -J_{2m} \\ \hline \vdots & \vdots & & \vdots \\ \hline -J_{m1} & -J_{m2} & \cdots & I_n - J_{mm} \end{array}\right]. \qquad (19)$$

In above formulas, I_n means n-th dimesional identity matrix , J_{pq} means

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

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 $J(\mathbf{k}_1^{(l}, \mathbf{k}_2^{(l)}, ..., \mathbf{k}_m^{(l)}) \rightarrow I_{mn}$ in case of $h \rightarrow 0$. Due to that property, DP-MP iterative refinement may be applicable to them for IRK method with small h, because they would be well-conditioned in many cases. In such situations, DP-MP iterative refinement may speed up IRK method for requiring MP approximation.

To confirm the hypothesis, we prepare the constant linear ODE

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

with the well-conditioned 128th dimentional matrix $A(\|A\|_1 \approx 10^3)$ based on 16), and then apply the 3 stages, 6th order Gauss type $L = 50$ decimal digits IRK method to it. Figure 8 shows the history of relative errors in the approximations vs stepsize h, and the histories of condition numbers and Frobenius norms of $J(\mathbf{k}_1^{(l)}, \mathbf{k}_2^{(l)}, ..., \mathbf{k}_m^{(l)})$.

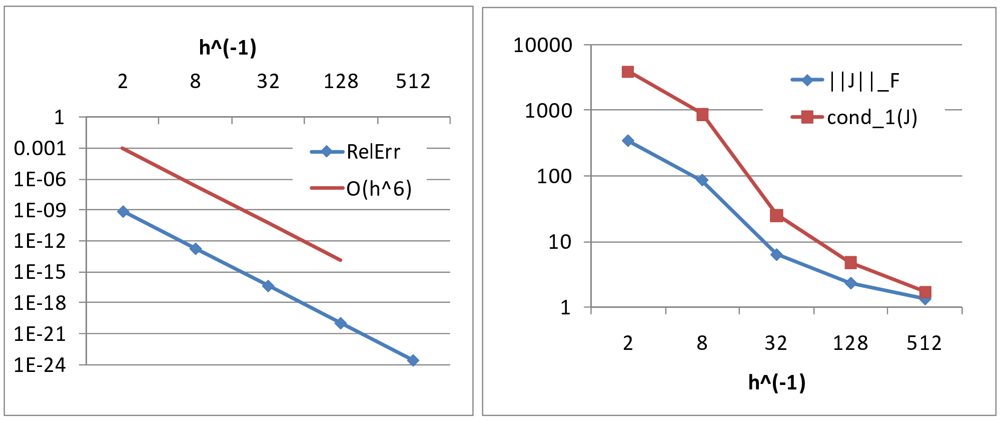

Figure 8. History of Maximum Relative Errors (Left), Frobenius Norms and Condition Numbers of J (Right)

In case of $h = 512$, we can recognize that about 24 decimal digits approximations are obtained and that the condition numers decrease in 3.8×10^3 to 1.7. These facts verify the applicability of DP-MP iterative refinements to the linear ODE. We show the speedup ratio of IRK method with DP-MP iterative refinement versus with direct method in Figure 9.

IRK method with DP-MP iterative refinement can perform about 7.5 to 8.9 times (with GotoBLAS) better than with direct method. In case of with BNCpack, it can do about 3.8 to 4.8 times better.

6. Conclusion and Future Works

Numerical experiments we mentioned above clarify that all types of mixed precision iterative refinements can bring us the approximation at the same level of relative error as L decimal digits direct method if the sufficient conditions for convergence are satisfied, that well-tuned LAPACK like GotoBLAS or ATLAS enable us to perform DP-MP iterative refinement as well as SP-DP type one in comparison with direct method, and that the application to fully implicit Runge-Kutta methods may provide us the efficiency.

Our future works are to investigate mixed precision refinement method's applicability to more general class of initial value problems for ordinary differential equations through various numerical experiments. Especially the most important problems are time-dependent partial differential equations. In many cases, the discretization for these PDEs lead to ODEs with

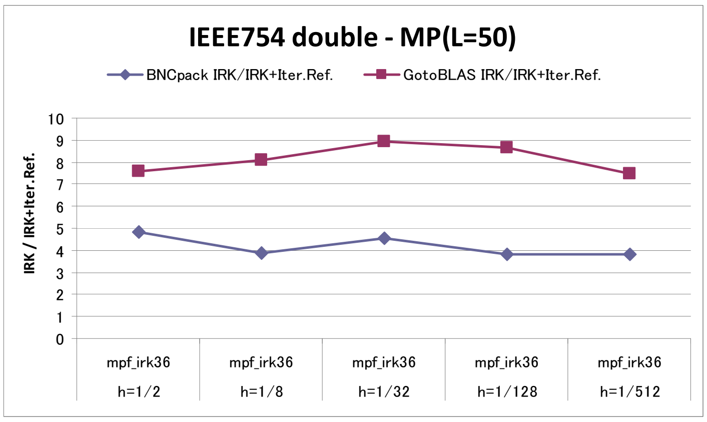

Figure 9. Speedup ratio of Imlicit Runge-Kutta Method (50 Decimal Precision): DP-MP vs Direct Method

sparse matrices, so iterative methods such as Krylov subspace methods will be more expected than direct methods as we experimented in this paper. Buttari et. al have already proved that SP-DP type iterarive refinement method can expand the performance of Krylov subspace methods. According to their results, we can expect more highly performance for DP-MP and MP-MP iterative refinements applied to Krylov subspace methods.

References

Swox AB. GNU MP. `http://gmplib.org/`.

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.

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.

S.P.Nørsett E.Hairer and G.Wanner. *Solving Ordinary Differential Equations*. Springer-Verlarg, 1996.

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

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

LAPACK. `http://www.netlib.org/lapack/`.

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

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