Research on the Convergence of Iterative Method Using Mixed Precision Calculation Solving Complex Symmetric Linear Equation

Koki Masui (桝井晃基)[1], Masao Ogino (荻野正雄)[2]

[1]Graduate School of Informatics, Nagoya University, Nagoya, Aichi 464, Japan
[2]Faculty of Informatics, Daido University, Nagoya, Aichi 457, Japan

This study investigates the complex symmetric linear equations that appear in high-frequency electromagnetic field analysis. As an efficient linear solver, we propose a mixed-precision iterative method with double- and double-double (DD)-precision floating-point numbers and an efficient implementation of DD-precision arithmetic with fused multiply-add instructions. Using the proposed method, we successfully reduce both the iteration count and calculation time compared with the conventional method. Moreover, we demonstrate the relation between multiple-precision arithmetic and the acceleration factor of the incomplete Cholesky factorization.

***Index Terms*—Complex symmetric matrices, Electromagnetic fields, High-precision calculation, Iterative methods.**

I. Introduction

In solving the complex symmetric linear equations derived from the edge finite element analysis of high-frequency electromagnetic fields [1], iterative methods suffer from oscillating residual norm histories and slow convergence rates. Various iterative methods and preconditioners have been developed [2], [3], but these issues remain unsolved. Furthermore, as the coefficient matrix of the system to be solved becomes ill-conditioned, even if an iterative method is stopped at a sufficiently small error of the residual, the obtained approximate solution may differ significantly from the true solution. To reduce the number of iterations require to obtain a solution with the desired precision, we focus on multiple-precision calculations, which have higher precision than double-precision floating-point numbers. However, arithmetic operations on complex numbers using multiple-precision floating-point numbers require several times the calculation costs of double-precision numbers [4]. Therefore, we propose an efficient implementation of multiple-precision arithmetic and develop a mixed precision iterative method for double- and higher-precision numbers.

II. High-Precision Calculations

A. DD-precision complex numbers

In this study, we adopt double-double (DD)-precision numbers in our multiple-precision arithmetic. First proposed by Bailey [5] and Briggs [6], DD is a software-based multiple-precision floating-point arithmetic that uses approximately quadruple-precision floating-point numbers of 106 bits in the form of two double-precision floating-point numbers. DD regards the two doubles as higher and lower bits, respectively. For instance, the QD library [7] provides DD precision.

Let us consider how to implement arithmetic operations on complex numbers with DD precision. DD-precision complex numbers require two double-precision numbers for both the real part and the imaginary part, and the ability to perform complicated operations with them. The C language's *double_Complex* type can perform complex arithmetic with single- or double-precision floating-point numbers, similar to operator overloading in C++, but this is not compatible with DD-precision numbers. Secondly, the complex class template in C++ can be combined with the QD library and DD-precision complex number arithmetic can be used with operator overloading. However, this results in poor calculation performance in our experience, and it is difficult to improve the performance. Therefore, we propose an effective implementation of DD-precision complex numbers.

```
1   dd_comp ddc_mult(dd_comp a, dd_comp b){
2    dd_comp c;
3    double d1, d2, d3, d4;
4    //c.re = a.re * b.re  −  a.im * b.im;
5    d1 = a.re.hi  *  b.re.hi;
6    d2 = fma(a.re.hi, b.re.hi, −d1);
7    d2 = fma(a.re.hi, b.re.low, d2);
8    d2 = fma(a.re.low, b.re.hi, d2);
9    d3 = a.im.hi  *  b.im.hi;
10   d4 = fma(a.im.hi, b.im.hi, −d1);
11   d4 = fma(a.im.hi, b.im.low, d2);
12   d4 = fma(a.im.low, b.im.hi, d2);
13
14   c.re = twosum(d1, −d3);
15   c.re.low  +=  d2 − d4;
16   c.re  =  fasttwosum(c.re.hi, c.re.low);
17
18  //c.im = a.re * b.im +  a.im * b.re;
19   d1 = a.re.hi  *  b.im.hi;
20   d2 = fma(a.re.hi, b.im.hi, −d1);
21   d2 = fma(a.re.hi, b.im.low, d2);
22   d2 = fma(a.re.low, b.im.hi, d2);
23   d3 = a.im.hi  *  b.re.hi;
24   d4 = fma(a.im.hi, b.re.hi, −d1);
25   d4 = fma(a.im.hi, b.re.low, d2);
26   d4 = fma(a.im.low, b.re.hi, d2);
27
28   c.im  =  twosum(d1, d3);
29   c.im.low  +=  d2  +  d4;
30   c.im  =  fasttwosum(c.re.hi, c.re.low);
31
32   return c;}
```

Fig. 1. Effective implementation of DD-precision complex number multiplication.

Manuscript received August 19, 2019; revised October 21, 2019; accepted XXXX XX, 2019. Date of publication XXXX XX, 2019; date of current version XXXX XX, 2019.

Koki Masui (e-mail: masui@hpc.itc.nagoya-u.ac.jp)

calculate preconditioning matrix M_D^{-1}, initial guess $\mathbf{x}_D^0$
$\mathbf{r}_D^0 = \mathbf{b}_D - A_D\mathbf{x}_D^0$, $\mathbf{z}_D^0 = M_D^{-1}\mathbf{r}_D^0$, $\mathbf{p}_D^0 = \mathbf{z}_D^0$
for $n = 0,1,2 \ldots$ until $\|\mathbf{r}_D^n\|/\|\mathbf{r}_D^0\| \le \varepsilon$
  $\mathbf{q}_D^n = A_D\mathbf{p}_D^n$
  $\alpha_{DD} = (\bar{\mathbf{r}}_D^n, \mathbf{z}_D^n)_{DD}/(\bar{\mathbf{p}}_D^n, \mathbf{q}_D^n)_{DD}$
  $\mathbf{x}_D^{n+1} = \mathbf{x}_D^n + \alpha_D\mathbf{p}_D^n$
  $\mathbf{r}_D^{n+1} = \mathbf{r}_D^n - \alpha_D\mathbf{q}_D^n$
  $\mathbf{z}_D^{n+1} = M_D^{-1}\mathbf{r}_D^{n+1}$
  $\beta_{DD} = (\bar{\mathbf{r}}_D^{n+1}, \mathbf{z}_D^{n+1})_{DD}/(\bar{\mathbf{r}}_D^n, \mathbf{z}_D^n)_{DD}$
  $\mathbf{p}_D^{n+1} = \mathbf{z}_D^{n+1} + \beta_D\mathbf{p}_D^n$
end for

Fig. 2. Mixed precision COCG method for solving $A\mathbf{x} = \mathbf{b}$.

Firstly, for comparison, we defined a double-precision complex data type using C structures, named *double_comp*, which realizes double-precision complex numbers and contains two double-precision variables, *re* and *im*, for the real and imaginary parts, respectively.

Next, we defined a DD-precision complex data type using C structures, named *dd_comp*, to realize DD complex numbers. This data type contains two structures, *re* and *im*, for the real and imaginary parts, respectively. Moreover, each part has two double-precision variables, *hi* and *low*, for higher and lower bits in DD, respectively. For code optimization, we used fused multiply-add (FMA) instructions to reduce the cost of multiplying and dividing of complex numbers.

Fig. 1 shows the pseudocode of the proposed multiplication process for DD-precision complex numbers using *dd_comp* type. In this figure, *fma* refers to FMA instructions, *twosum* [8] and *fasttwosum* [9] denote the error-free transformation of the sum of two floating-point numbers. When we use traditional DD-precision complex numbers, there are 58 arithmetic instructions, including FMA instructions, required to multiply DD-precision complex numbers. Our proposed efficient implementation requires 38 arithmetic instructions, including FMA instructions.

B. Mixed-Precision Iterative Methods

As DD numbers have several times the calculation costs of double-precision numbers, we consider the partial use of DD. In solving a complex symmetric linear equation $A\mathbf{x} = \mathbf{b}$ using the conjugate orthogonal conjugate gradient (COCG) method, our numerical experiments indicate that the accuracy with which α and β are calculated is crucial to satisfying the conditions $\bar{\mathbf{r}}^{iT}\mathbf{r}^j = 0$ and $\bar{\mathbf{p}}^{iT}A\mathbf{p}^j = 0$ for $i \neq j$, where $\mathbf{r}^i$ is the i-th residual and $\mathbf{p}^i$ is the i-th search direction. Therefore, as a partial usage technique of multiple-precision arithmetic, we propose a mixed-precision COCG method. An algorithm to solve $A\mathbf{x} = \mathbf{b}$ is shown in Fig. 2. The subscripts *D* and *DD* represent double-precision numbers and DD numbers, respectively, ε is the convergence criterion, and $(\mathbf{x}, \mathbf{y})$ denotes $\sum \bar{x}_i y_i$. Note that we only apply DD-precision arithmetic for the summation in the dot product. $(\mathbf{x}, \mathbf{y})_{DD}$ denotes that the products of corresponding components are calculated in double precision and the sum of the result is in DD precision. After calculating α_{DD} and β_{DD} in DD precision, the upper bits of these values are used as α_D and β_D in double precision.

III. Condition Number and Solution Error

Consider the following equation:

$$A\mathbf{x} = \mathbf{b}, \tag{1}$$

where A is a complex symmetric $n \times n$ matrix, $\mathbf{b}$ is a known vector, and $\mathbf{x}$ is an unknown vector. The condition number $\kappa(A)$ of the coefficient matrix is given by

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|. \tag{2}$$

Moreover, using the true solution $\mathbf{x}$ and an approximate solution $\tilde{\mathbf{x}}$ obtained by the iterative method, the following equation holds:

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} \le \kappa(A)\frac{\|\mathbf{b} - A\tilde{\mathbf{x}}\|}{\|\mathbf{b}\|}. \tag{3}$$

From (3), if the condition number $\kappa(A)$ is large, even if the relative residual $\|\mathbf{b} - A\tilde{\mathbf{x}}\|/\|\mathbf{b}\|$ is sufficiently small, the relative error of the solution will not necessarily be small. In other words, it is necessary to consider the accuracy of the approximate solution obtained by numerical calculations in ill-condition problems. Although the convergence criterion is often determined based on experience, the accuracy of the approximate solution is not guaranteed. To obtain an approximate solution with sufficient accuracy, for example, we have to set the convergence criterion smaller. However, if we use a smaller convergence criterion in consideration of this issue, the calculation costs will be unnecessarily increased. Therefore, it is important to understand the relation between the solution error and the relative norm. We aim to evaluate these effects through numerical experiments.

IV. Preconditioning and Iterative Method

The incomplete Cholesky (IC) factorization [10], symmetric successive over-relaxation (SSOR) [11], and diagonal scaling are widely used as preconditioners for (1). While diagonal scaling and SSOR have the advantage of requiring less memory, the use of an acceleration factor (AF) with IC factorization often results in better convergence in the edge finite element analysis of electromagnetic fields. Therefore, we use the IC preconditioner with AF in this study. As the AF, a value slightly larger than 1.0 is typically used. Moreover, an automatic determination method [12], [13] has been reported. The optimal AF for a mixed-precision iterative method is not clear. Thus, in this study, we investigate the optimal AF value through numerical experiments.

V. Numerical Experiments

As a high frequency electromagnetic field problem, let us consider a wave equation having an electric field **E** derived

from Maxwell's equation as an unknown [1]:

$$\int_\Omega \mathrm{rot}\mathbf{E}_h \cdot \mu^{-1}\mathrm{rot}\mathbf{E}_h^* d\Omega - \int_\Omega \left(\omega^2\varepsilon' - i\omega\sigma\right)\mathbf{E}_h \cdot \mathbf{E}_h^* d\Omega = i\omega \int_\Omega \mathbf{J}_h \cdot \mathbf{E}_h^* d\Omega, \quad (4)$$

where μ (H/m) is the permeability, ϵ_0 (F/m) is the permittivity of a vacuum, $\epsilon'$is the relative permittivity, ω (rad/s) is a single angular frequency, i is the imaginary unit, $\mathbf{J}_h$ (A/m^2) is the current density, and σ (S/m) is the conductivity. To solve (4), we used the edge finite element method and obtained a complex symmetric linear system.

To evaluate the computational performance, numerical experiments were performed. The coefficient matrix A was obtained from TEAM Workshop Problem 29 with a phantom model [14], as shown in Fig. 3. The electrical constants of the phantom were set to be ϵ' = 80.0 and σ = 0.52. We constructed a medium-scale problem with 134,573 degrees of freedom. The number of nonzero elements in the coefficient matrix was 2,123,849.

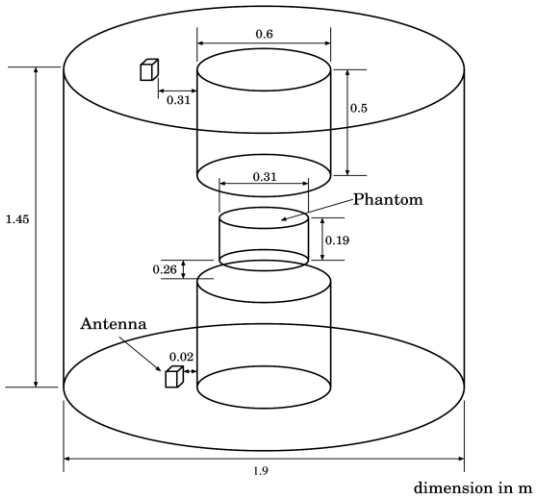

Fig. 3. Test model of TEAM Workshop Problem 29.

To construct the complex symmetric linear equation, we used ADVENTURE_Magnetic, provided by the ADVENTURE project [15]. All calculations were performed using an Intel CPU Core i7-7700 (3.60 GHz) processor with 64 GB memory, the gcc 7.3.0 compiler, and the "-O3 -mfma" optimization flag.

Let $A\mathbf{x} = \mathbf{b}$ be the linear equation to be solved, where $\mathbf{b}$ is given by computing $\mathbf{b} = A\mathbf{x}$ with $\mathbf{x} = (1.0 + 1.0i, \ldots, 1.0 + 1,0i)^T$ and the initial guess $\mathbf{x}^0$ is zero. To solve this problem, we used the COCG method with IC preconditioning.

Firstly, we examined the convergence history of the relative residual norm $\|\mathbf{r}^n\|_2/\|\mathbf{r}^0\|_2$, relative preconditioned residual norm $\|\mathbf{z}^n\|_2/\|\mathbf{z}^0\|_2$, and maximum error with respect to the true solution $\|\mathbf{x}^n - \mathbf{x}\|_\infty$ until the relative residual norm became 10^{-12} or less. In this case, the AF was 1.1. From Fig. 4, the relative residual norm differs greatly from the error norm because of the ill-conditioned coefficient matrix. The relative preconditioned residual norm is closer to the error norm, but there is still a gap. Therefore, we must set another 10^{-3} or smaller convergence criterion for the required accuracy. To calculate such small values, a high-precision calculation is needed.

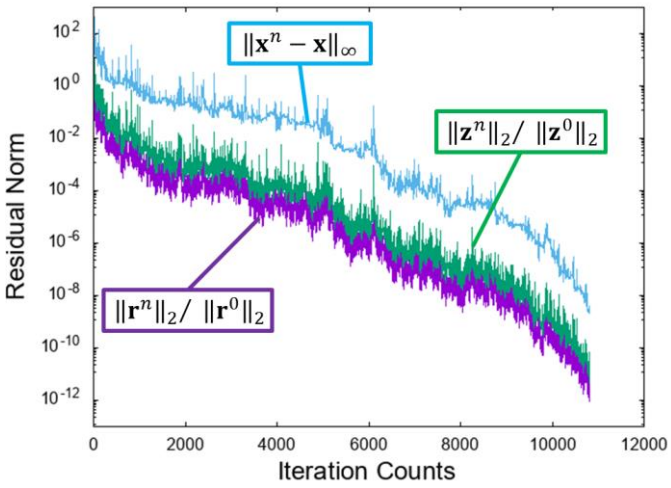

Fig. 4. Convergence history using DD precision.

Secondly, we examined the effect of AF on the iterative method with different precision arithmetic. The convergence criterion was set to $\|\mathbf{r}^n\|_2/\|\mathbf{r}^0\|_2 < 10^{-6}$. Figs. 5 and 6 show the iteration counts and calculation times as the AF of IC preconditioning was varied from 1.0–1.2 in increments of 0.01. DD indicates that all calculations were performed in DD precision. DD further reduced the number of iterations compared to double precision. In particular, in the case of AF = 1.0, DD reduced the number of iterations by 87% and reduced the calculation time by 43%. When AF was greater than 1.0, although it took several times longer than double-precision, DD reduced the number of iterations by a factor of about two-thirds.

In all cases, we succeeded in reducing the iteration count and calculation time compared with double precision by using mixed-precision calculations. Therefore, our mixed-precision system is effective for this problem. Thus, the iterative method using high-precision calculations is a robust method, and our proposed mixed-precision system can be faster than a double-precision system.

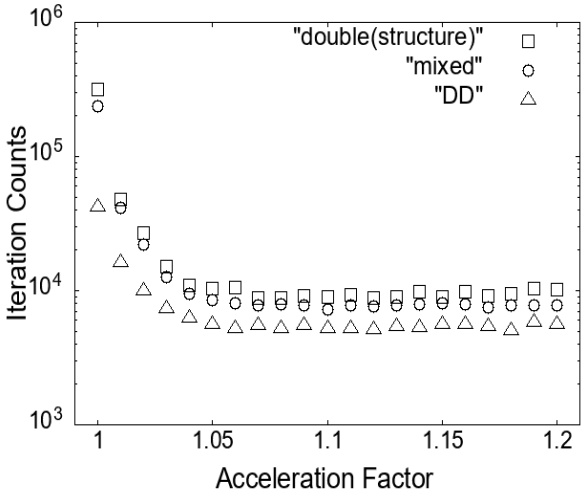

Fig. 5. Acceleration factor vs. iteration count.

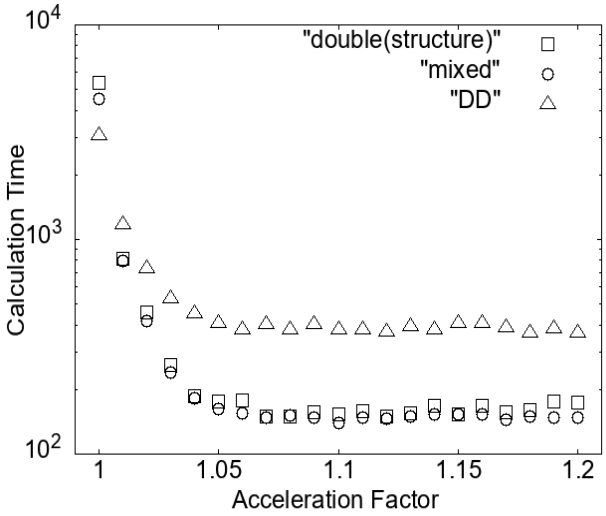

Fig. 6. Acceleration factor vs. calculation time.

Table 1 presents the computational performance in the fastest case with the optimized acceleration factor. In the table, double(_Complex) denotes the use of the default complex type in C. From Figs. 5 and 6, the optimum AF values varied slightly depending on the precision. The convergence histories of double(_Complex) and our double-precision complex type using the (double_comp) structure are almost the same, although our method reached the solution faster than double(_Complex). DD reduced the number of iterations compared with the double-precision case, but required more time than double(_Complex). However, our proposed system successfully reduced both the iteration count and calculation time.

TABLE 1
COMPUTATIONAL PERFORMANCE IN FASTEST CASE

Precision	AF	Iteration	Time (sec)
double (_Complex)	1.07	8,159	202
double (double_comp)	1.07	8,194	153
mixed (proposed)	1.1	7,254	141
DD	1.18	5,081	387

Finally, Table 2 gives the computational performance in the fastest case using the right-hand-side vector of (4). The convergence criterion was $\|\mathbf{r}^n\|_2/\|\mathbf{r}^0\|_2 < 10^{-9}$ in this numerical experiment. The optimum AF values are almost the same. This table indicates that, even in this case, our proposed system reduced both the iteration count and calculation time. These results demonstrate the effectiveness of our proposed system when applied to realistic problems.

TABLE 2
COMPUTATIONAL PERFORMANCE USING THE RIGHT-HAND-SIDE VECTOR

Precision	AF	Iteration	Time (sec)
double (_Complex)	1.08	6,314	140
double (double_comp)	1.07	6,276	103
mixed (proposed)	1.07	5,921	98
DD	1.06	4,444	291

VI. CONCLUSION

In this research, we derived an efficient implementation of DD-precision complex-number arithmetic. This paper described the implementation of a mixed-precision COCG method and reported the results of numerical experiments to solve the complex symmetric linear equations that appear in high-frequency electromagnetic field analysis using the edge finite element method.

The following findings were obtained by numerical experiments using the developed system.

- The relative residual norm may differ greatly from the error norm in high-frequency electromagnetic field analysis with the edge finite element method.
- DD-precision calculations can reduce the number of iterations of the iterative method, regardless of the AF, but the calculation time may be several times longer.
- Our proposed mixed-precision approach can reduce both the number of iterations and the computation time compared to double precision in many cases.

In future work, we will verify the effectiveness of the proposed method by applying it to larger and more complex problems. In addition, we aim to realize further acceleration by examining high-performance computing technology using Intel AVX instructions.

REFERENCES

[1] A. Takei, S. Yoshimura, H. Kanayama, "Large-Scale Parallel Finite Element Analyses of High Frequency Electromagnetic Field in Commuter Trains", *Comput. Modeling Engrg. Sci.*, vol.31, pp.13-24, 2008.
[2] H.A. van der Vorst and J.B.M. Melissen, "A petrov-galerkin type method for solving Ax=b and where A is symmetric complex", *IEEE Trans. Magn.*, vol.26, pp.706-708, 1990.
[3] K. Fujiwara, T. Nakata, H. Fusayasu, "Acceleration of convergence characteristic of the ICCG method", *IEEE Trans. Magn.*, vol.29, no.2, pp.1958-1961, 1993.
[4] K. Masui, M. Ogino, "Performance evaluation of multiple-precision conjugate orthogonal conjugate gradient method in complex symmetric linear equations", *Transactions of JSCES*, p. 20180007, 2018. (*in Japanese*)
[5] Bailey, D.H, http://crd-legacy.lbl.gov/~dhbailey/mpdist/
[6] Briggs, K., http://keithbriggs.info/doubledouble.html
[7] High-Precision Software Directory, http://crd-legacy.lbl.gov/~dhbailey/mpdist/
[8] D.E. Knuth, *The Art of Computer Programming, Volume 2, Seminumerical Algorithms*, Addison Wesley, Reading, Massachusetts, 1969.
[9] T.J. Dekker, "A floating-point technique for extending the available precision", *Numer. Math.*, Vol. 18, pp. 224-242, 1971.
[10] J. A. Meijerink and H. A. van der Vorst, "An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix", *Mathematics of Computation*, Vol. 31, No. 137, pp. 148-162, 1977
[11] O. Axelsson, "A generalized SSOR method", *BIT Numerical Mathematics*, vol.13, pp 443 – 46, 1972.
[12] A. Takada, S. Noguchi, and H. Igarashi, "A New Acceleration Factor Decision Method for ICCG Method Based on Condition Number", *IEEE Trans. Magn.*, Vol. 48, No. 2, pp. 519-522, 2012.
[13] J. Kitao, Y. Takahashi, K. Fujiwara, T. Mifune, and T. Iwashita, "Automatic Determination of Acceleration Factor Based on Residual and Functional in Shifted ICCG Method for 3-D Electromagnetic Field Analyses", *IEEE Trans. Magn.*, Vol. 49, pp. 1741-1744, 2013
[14] Y. Kanai, "Description of TEAM Workshop Problem 29: Whole body cavity resonator", *TEAM Workshop in Tucson*, 1998.
[15] ADVENTURE Project, https://adventure.sys.t.u-tokyo.ac.jp/