IC($\boldsymbol{p}$) Preconditioning with Acceleration Factor for High-Frequency Electromagnetic Field Analysis

Koki MASUI (Osaka University) Masao OGINO (Daido University)
Takahiro KATAGIRI (Nagoya University) Fumihiko INO (Osaka University)

Backgrounds and objectives

- Demand for speed-up of the iterative method for solving complex symmetric linear equations derived from edge finite element analysis of high-frequency electromagnetic fields
  - ✓ poor convergence due to ill-conditioned coefficient matrix
- IC(p) can improve the convergence (p: fill-in level)
  - ✓ There are few examples
  - ✓ An efficient implementation method has not been established

- <u>Efficient implementation of iterative methods for speeding up</u>
  - using Conjugate Orthogonal Conjugate Gradient (**COCG**) method
  - using **IC($\boldsymbol{p}$) preconditioning** for complex symmetric matrix

IC($\boldsymbol{p}$) preconditioning w/ fill-in

IC(0): non-zero element positions of preconditioning and coefficient matrix are the same
IC(p): generates the preconditioning matrix with additional non-zero element positions depending on the value of p.

The additional non-zero element positions are determined by

$$Level_{i,j} = \min\{Level_{i,j}, \min_{1\le k\le \min(i,j)}\{Level_{i,k} + Level_{k,j} + 1\}\}$$

```
L_{0,0} := 1.0
D_0 := A_{0,0} × α
for i = 1,2,3,…,n−1
  for j = 0,1,2,…,i
    if (i = j) then
      L_{i,j} := 1.0
    else
      if (Level_{i,j} = 0) then
        L_{i,j} := (A_{i,j} − Σ_{k=1}^{j−1} L_{i,k} D_k L_{j,k}) / D_j
      else if (Level_{i,j} ≤ p) then
        L_{i,j} := − Σ_{k=1}^{j−1} L_{i,k} D_k L_{j,k} / D_j
      end if
    end if
  end for
  D_i := A_{i,i} × α − Σ_{k=1}^{i−1} L_{i,k}^2 D_k
end for
```

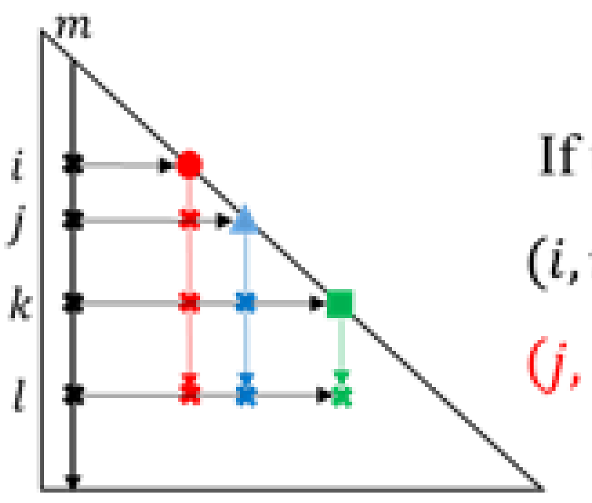
Algorithm of IC($\boldsymbol{p}$) preconditioning w/ fill-in

Efficient IC(p) preconditioning in COO-format

If we calculate the preconditioner as per the formula, the amount of calculation is $O(n^3)$.
In this study, we calculate the preconditioner matrix using the following algorithm.

- **<u>This method focuses on non-zero element positions in COO format.</u>**

1. Convert CRS-format coefficient matrix to COO format
2. Sort the coefficient matrix in Column-Major
3. Search and add preconditioners using the method shown in the figure below.
4. Removed overlapping parts of additional element positions
5. Sort preconditioner by Row-Major in ascending order
6. Convert the generated COO format preconditioner to CRS format



If the nonzero elements in column m are $(i,m), (j,m), (k,m)$, and (l,m), add new non-zero to $(j,i), (k,i), (l,i), (k,j), (l,j)$, and (l,k).

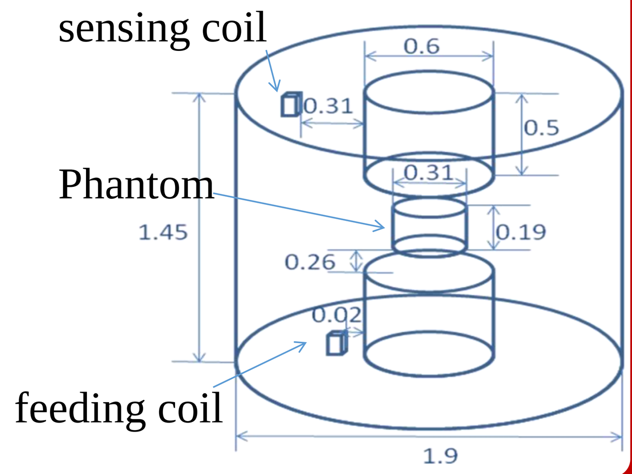
Numerical model

- Edge finite element equation for vector wave equation

$$\int_\Omega \mathrm{rot}\mathbf{E}_h \cdot \mu^{-1}\mathrm{rot}\mathbf{E}_h^* d\Omega - \int_\Omega (\omega^2\varepsilon' - j\omega\sigma)\mathbf{E}_h \cdot \mathbf{E}_h^* d\Omega = j\omega \int_\Omega \mathbf{J}_h \cdot \mathbf{E}_h^* d\Omega$$

where $\mathbf{E}_h$ (V/m) is FE approximation of electric field, $\mathbf{E}_h^*$ (V/m) test function satisfying $\mathbf{E}_h^* \times n = 0$ on $\partial\Omega$, $\mathbf{J}_h$ (A/m^2) FE approximation of current density, μ (H/m) permeability, ε' (F/m) real part of complex permittivity, σ (S/m) electric conductivity, $\omega = 2\pi f$ (rad/s) single angular frequency, f (Hz) frequency, n outward normal vector on the boundary, and j imaginary unit.

- TEAM Workshop Problem 29
  - # of rows, columns: 134,573, 439,176, 979,464
  - # of nonzeros: 2,123,849, 7,036,670, 15,794,744
  - $\varepsilon' = 80.0$ (F/m)
  - $\sigma = 0.52$ (S/m)



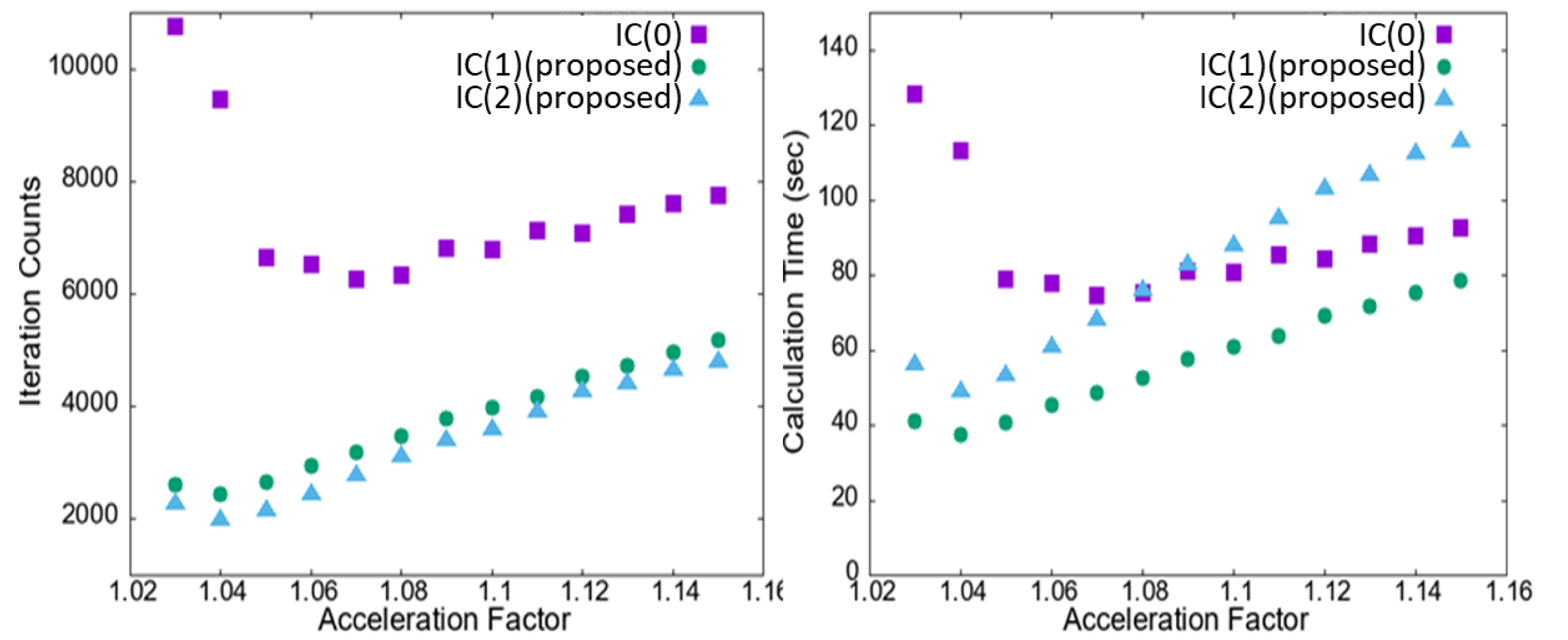
Numerical experiments

- Computer
  - CPU (clock frequency): Intel Core i9-9900 (3.6 GHz)
  - Compiler (flag): GCC-10.2.0 (-O3 -mfma)
- Solver
  - COCG method with shifted IC(p) preconditioner(p = 0, 1, 2)
  - Convergence criteria: $\varepsilon = 10^{-9}$

The number of nonzero elements in the preconditioner

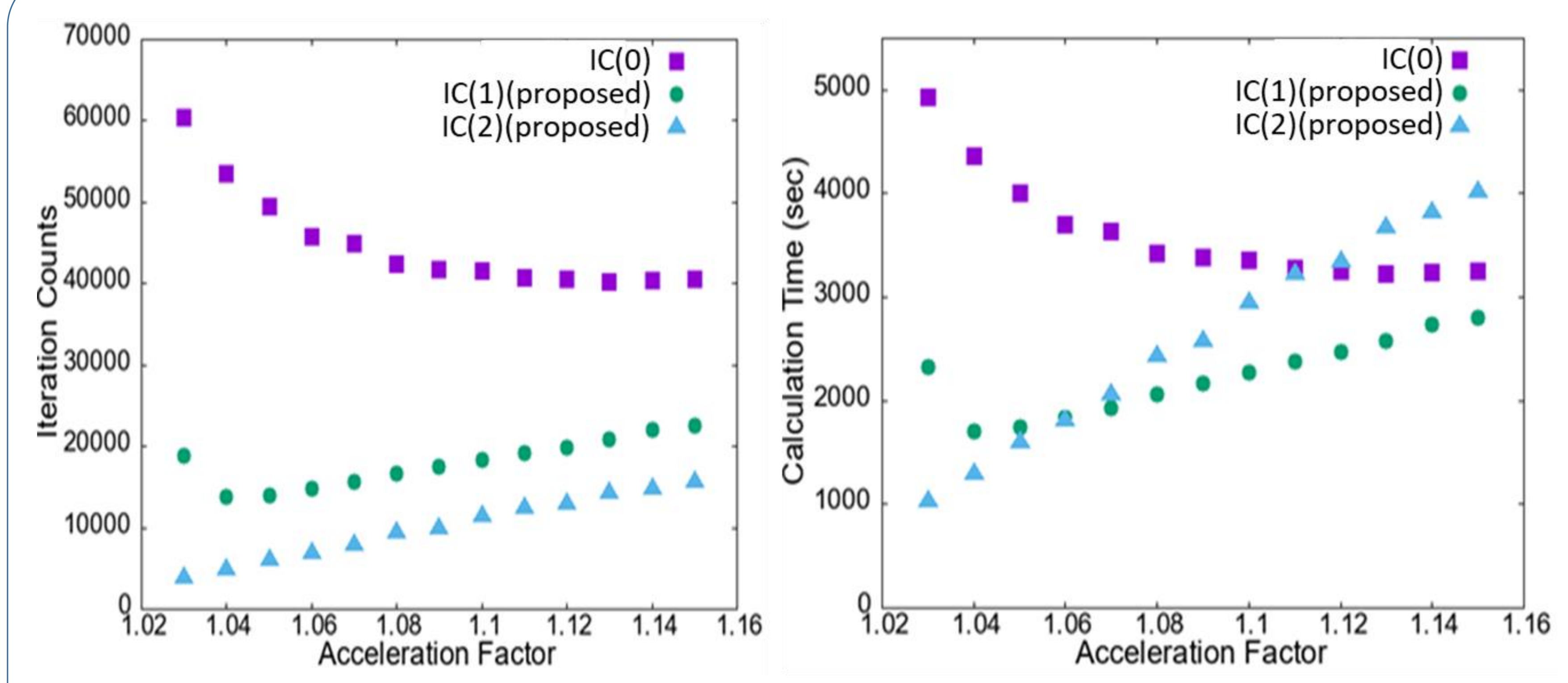
preconditioner＼DoF	134,573	439,176	979,464
IC(0)	1,129,211	3,737,923	8,387,104
IC(1)	2,094,070	9,853,214	22,268,605
IC(2)	4,680,924	27,919,791	63,799,785

134,573



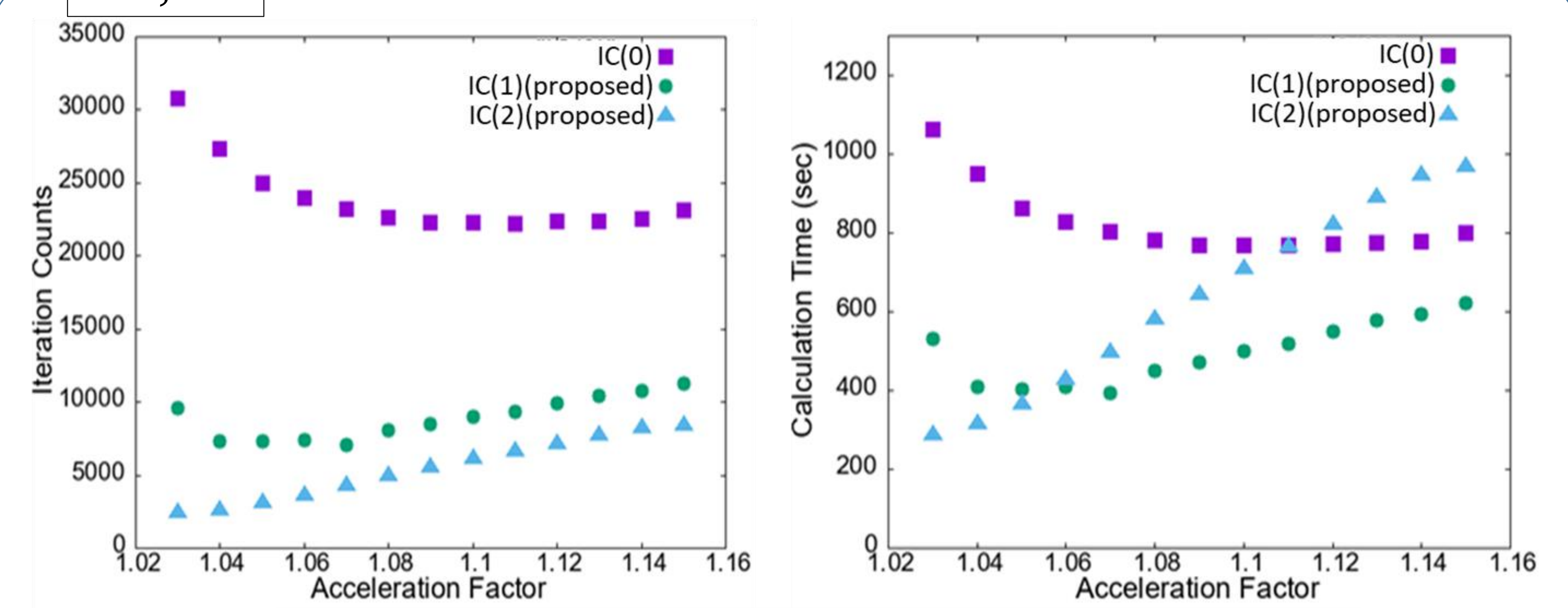
IC(1) and IC(2) improve the convergence.
Although IC (2) reduced the number of iterations the most, **IC (1) reduced the calculation time the most.**

979,464



The tendency is the same as the case of 439,176.

439,176



IC(2) reduced both of the number of iterations and calculation time the most.
※Pay attention to the value of the acceleration factor

In all cases, Our proposed method system successfully reduced both iteration counts and calculation time

Concluding remarks and future works

- ✓ Proposed an efficient IC(p) preconditioning method for complex symmetric matrices
- ✓ Succeeded in reduction of iteration counts and calculation time by IC(p) with fill-in
  - Optimal fill-in level depends on the size of the problem

◆Future work

- ✓ Estimate the optimum acceleration factor
- ✓ Aim to realize further acceleration by examining high-performance computing technology using Intel AVX instructions