A Multilevel $\mathcal{H}$-matrix-based Approximate Matrix Inversion Algorithm for Vectorless Power Grid Verification

Wei Zhao, Yici Cai, Jianlei Yang

Tsinghua University

ASP-DAC 2013, Yokohama

January 22th

Outline

- **Introduction**
- **Proposed Approach**
  - **Algorithm Overview**
  - $\mathcal{H}$**-matrices**
  - **Multilevel Methods**
  - **Iterative Refinement Scheme**
- **Experimental Results & Summary**

Power Grid Verification

- Power grid verification is crucial for silicon success
- Simulation based approach
  - For the given current loadings $\mathbf{i}$, to obtain voltage noise by solving

    $$Gv = i \quad \text{(R Model)}$$

  - Simulation is not enough
    - Need to simulate large number of current vectors to cover usual working modes
    - Early stage verification cannot be performed since the detailed current waveform information is still unknown
    - No guarantee the worst noise (but not over pessimistic) can be found

Vectorless Power Grid Verification

- **Vectorless approach**
  - **Early stage verification technique**
  - **Optimization approach to obtain the worst case of IR-Drop**
- **Problem formulation**
  - **Given current constraints to specify the feasible space of current excitations**
    - **Local constraints** $\boldsymbol{0 \leq i \leq I_L}$
    - **Global constraints** $\boldsymbol{Ui \leq I_G}$
  - **To estimate the worst-case voltage fluctuations by solve optimization problems**

$$\boldsymbol{v = G^{-1} i}$$

Vectorless Power Grid Verification

- The problem can be divided into two major tasks
  - Let $c_i \triangleq G^{-1}e_i$

    where e_i is the $n \times 1$ vector of all zeros except the i-*th* component being 1, it is to obtain the i-*th* column of G^{-1} by solving $Gx = e_i$
  - The voltage of the i-*th* node can be obtained by

$$v_i = c_i^T i$$

  - Task 1: compute c_i by solving $Gx = e_i$
  - Task 2: maximize $v_i = c_i^T i$ s.t.

$$Ui \le I_G \text{ and } 0 \le i \le I_L$$

- Total cost to verify a power grid with N nodes
  - Solving linear equations with N unknowns for N times
  - Solving LP problems for N times

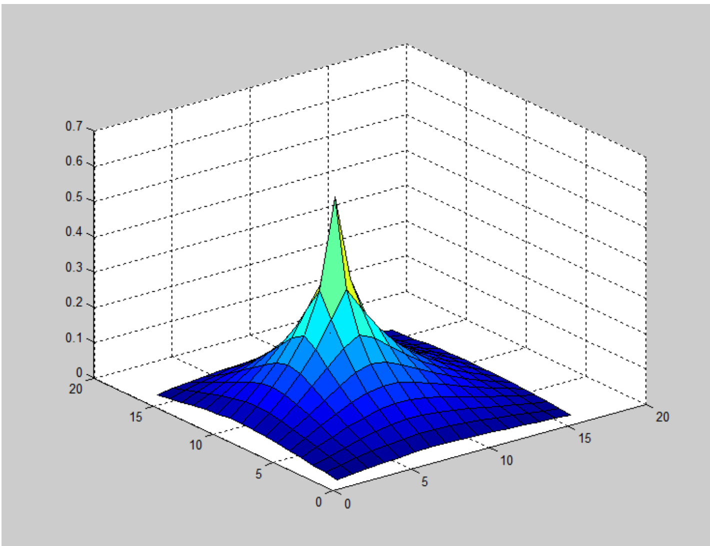
Task 1: More than 80% computation cost!

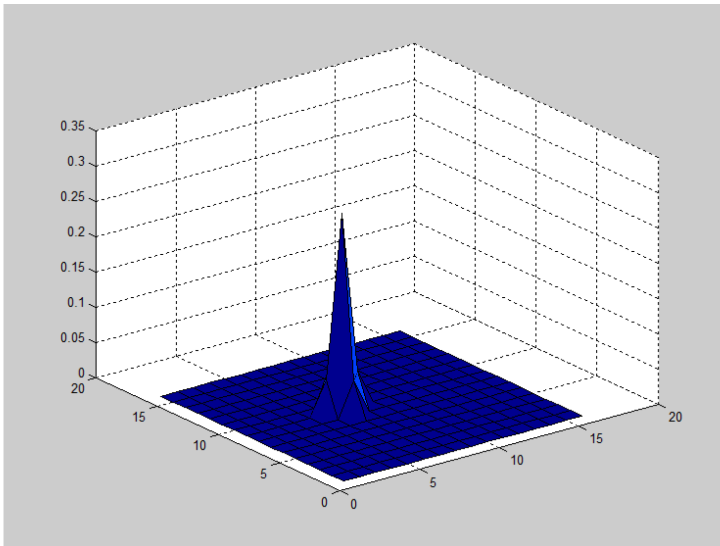
Related Works for Task 1: Acceleration

- Important observations
  - Multiple right-hand sides problem
    - Direct solvers are more favored to be adopted
  - Relatively lower accuracy requirement
    - Tradeoff between accuracy and solving efficiency
- Previous works - acceleration methods
  - Sparse Approximate Inverse
    - SPAI (N. H. Abdul Ghani and F. N. Najm, DAC 2009)
    - AINV (M. Avci and F. N. Najm, ICCAD 2010)
  - Hierarchical matrix inversion (X. Xiong and J. Wang, ICCAD 2010)

The Essence of Sparse Matrix Inverse

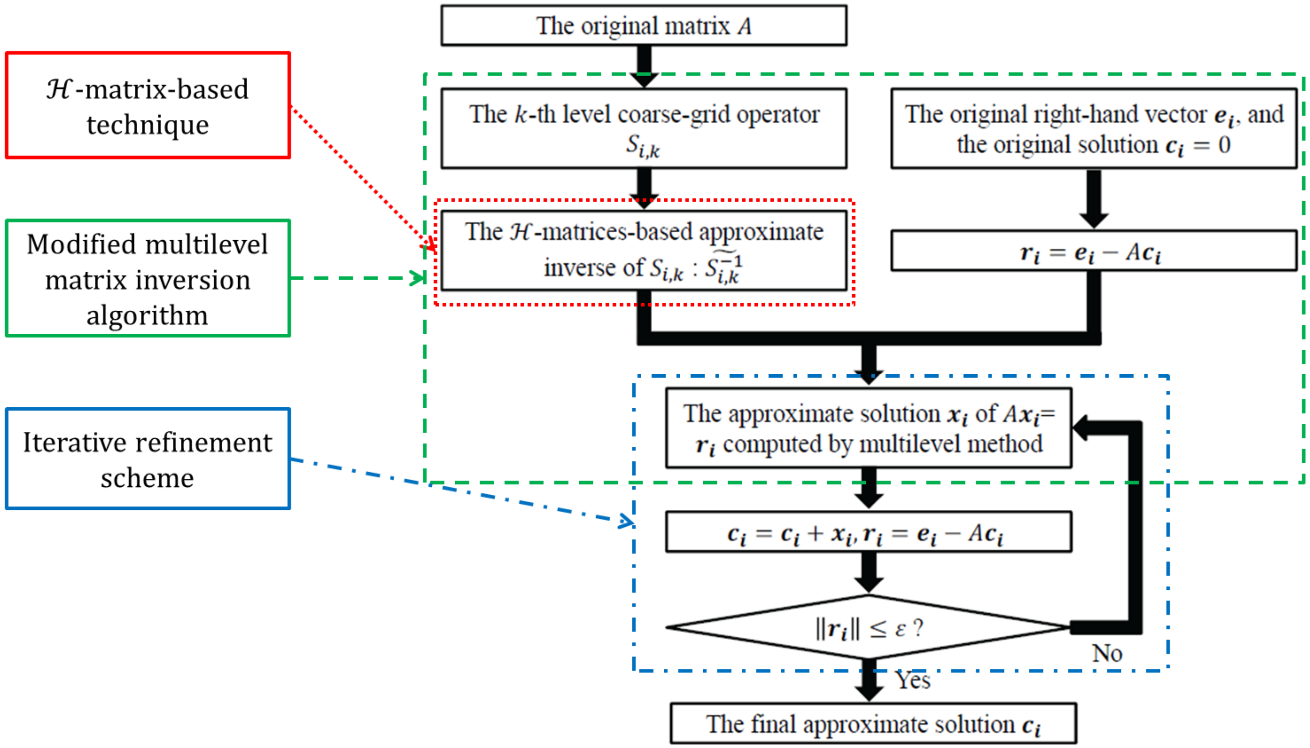
- Computing the sparse matrix inverse is equivalent to obtain the sensitivity of each node for all current variables
- The main difficulty for approximate inverse methods: global coupling property of the linear system
- If we want to get a better sparse approximation, we have to find a method which can bring in more global information with a certain amount of memory footprint.





Proposed Algorithm Framework

- **Major techniques used in the proposed algorithm**
  - $\mathcal{H}$-matrix-based technique
  - Modified multilevel matrix inversion algorithm
  - Iterative refinement scheme

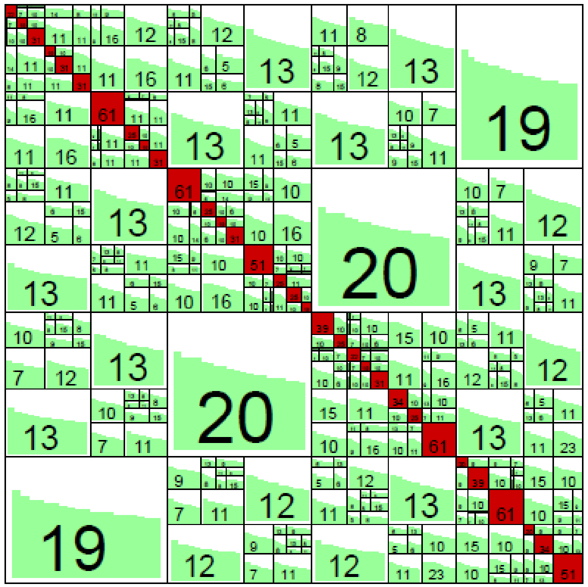


$\mathcal{H}$-matrices

- **Data-sparse representation**
  - **Main idea**
    - **Two parts of the geometry I and J: well separated (e.g. have a positive distance)**

      SPAI: the matrix block $M \in \mathbb{R}^{I \times J}$ is a zero matrix

      $\mathcal{H}$-matrix: the matrix block $M \in \mathbb{R}^{I \times J}$ can be approximated by a low-rank matrix
  - **Hierarchical block structure**
  - **Low-rank approximation**

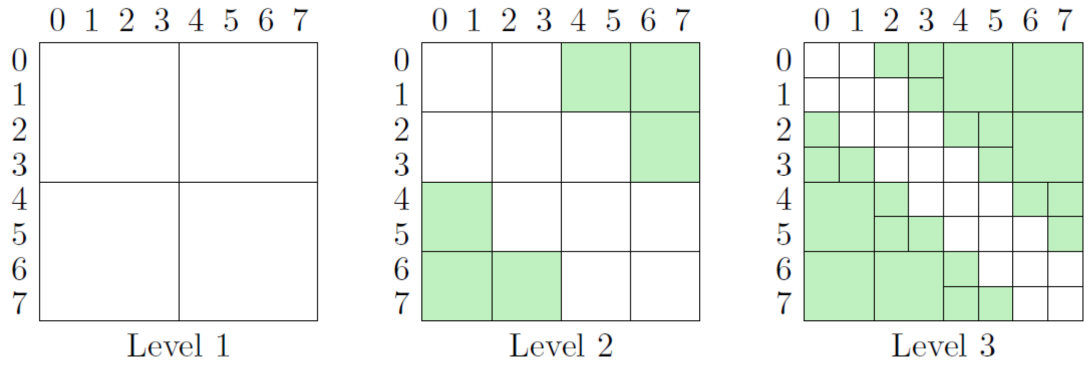


$\mathcal{H}$-matrices

- Time and space complexity: almost linear

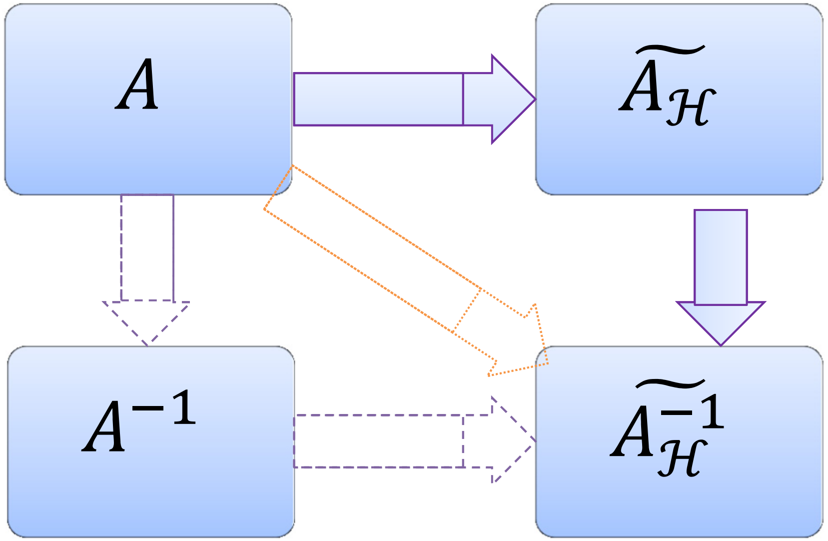
Operation	Complexity
Matrix Vector Product	$\mathcal{O}(n \log n)$
Matrix Addition	$\mathcal{O}(n \log n)$
Matrix Multiplication	$\mathcal{O}(n \log^2 n)$
Matrix Inversion	$\mathcal{O}(n \log^2 n)$
LU Factorisation	$\mathcal{O}(n \log^2 n)$

- $\mathcal{H}$-matrix construction
  - Cluster tree
    - Geometric clustering
    - Algebraic clustering
  - Block cluster tree



$\mathcal{H}$-matrices

- $\mathcal{H}$-matrix-based approximate inverse construction
  - Computation flow



  - Two choices
    - Direct $\mathcal{H}$-matrix inversion
    - $\mathcal{H}$-Cholesky factorization

Multilevel Methods

- **Block matrix inversion**
  - **2×2 block partitioned matrix:** $A = \begin{bmatrix} D_1 & B \\ B^T & D_2 \end{bmatrix}$
  - **The block LU factorization of** A**:**

$$A = \begin{bmatrix} D_1 & B \\ B^T & D_2 \end{bmatrix} = \begin{bmatrix} D_1 & 0 \\ B^T & S \end{bmatrix} \begin{bmatrix} I & D_1^{-1}B \\ 0 & I \end{bmatrix} \textbf{(}S = D_2 - B^T D_1^{-1} B\textbf{)}$$

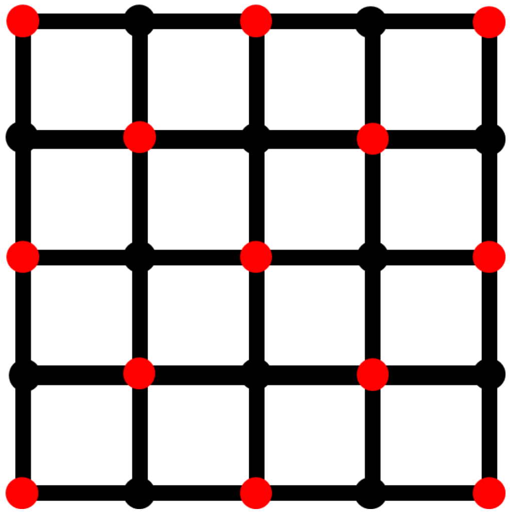
  - **The block forward and backward substitution:**

$$\begin{bmatrix} D_1 & 0 \\ B^T & S \end{bmatrix} \begin{bmatrix} I & D_1^{-1}B \\ 0 & I \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

1. $x_1 := D_1^{-1} b_1$
2. $x_2 := S^{-1}(b_2 - B^T x_1)$
3. $x_1 := x_1 - D_1^{-1} B x_2$

Multilevel Methods

- **Block matrix inversion**
  - **Red-black ordering**



  - D_1**: diagonal matrix**
  - **The Main problem: inverse of the Schur complement**

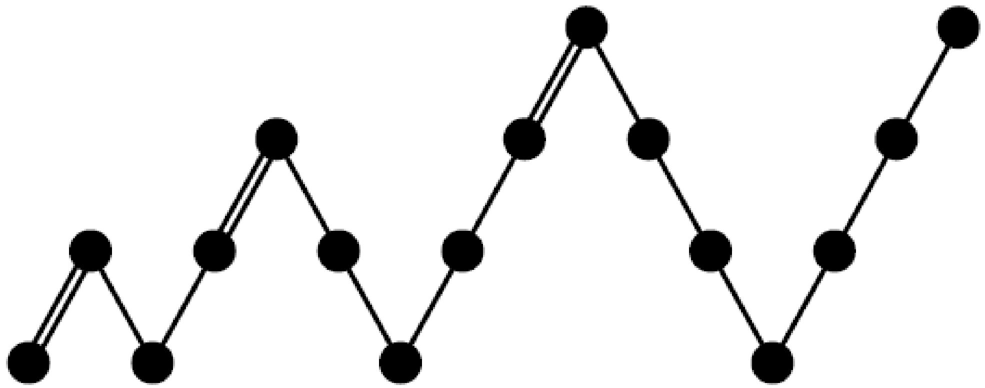
- **Approximate inversion**

1. $x_1 \coloneqq D_1^{-1} b_1$
2. Compute $M_S^{-1} \cong S^{-1}$
3. $x_2 \coloneqq M_S^{-1}(b_2 - B^T x_1)$
4. $x_1 \coloneqq x_1 - D_1^{-1} B x_2$

  - **The approximate inverse of the Schur complement can be computed by the $\mathcal{H}$-matrix-based approximate inverse method**

Multilevel Methods

- **Algebraic multigrid methods**
  - **Basic notation**
    - **Fine-grid operator** A^h
    - **Coarse-grid operator** A^{2h}
    - **Restriction operator** I_h^{2h}
    - **Prolongation operator** I_{2h}^h
  - **Main ideas**
    - **Coarse-grid correction**
    - **Nested iteration**



Multilevel Methods

Multigrid methods

- Fine-grid operator A^h
- Restriction operator I_h^{2h}
- Prolongation operator I_{2h}^h
- The coarse-grid operator $A^{2h} = I_h^{2h} A^h I_{2h}^h$
- Coarse-grid correction

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} D_1^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + W^T S^{-1} W \left(\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} - A \begin{bmatrix} D_1^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right)$$

- Nested iteration

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} D_1^{-1} & 0 \\ -D_2^{-1} B^T D_1^{-1} & D_2^{-1} \end{bmatrix} \begin{bmatrix} 0 & -B \\ 0 & 0 \end{bmatrix} W^T S^{-1} W \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} D_1^{-1} & 0 \\ -D_2^{-1} B^T D_1^{-1} & D_2^{-1} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Block matrix inversion

- The original matrix A
- $W = [-B^T D_1^{-1} \quad I]$
- $W^T = [-B^T D_1^{-1} \quad I]^T$
- The Schur complement $S = D_2 - B^T D_1^{-1} B$

Multilevel Methods

- **Approximate block matrix inversion**
  - **Algorithm based on coarse-grid correction**

1. $x_1 := D_1^{-1} b_1$
2. Compute $M_S^{-1} \cong S^{-1}$, $x_2 := M_S^{-1}(b_2 - B^T x_1)$
3. $x_1 := x_1 - D_1^{-1} B x_2$

  - **Modified algorithm based on nested iteration**

1. $x_1 := D_1^{-1} b_1$
2. $u := b_2 - B^T x_1$
3. Compute $M_S^{-1} \cong S^{-1}$, $x_2 := M_S^{-1} u$
4. $x_1 := x_1 - D_1^{-1} B x_2$
5. $x_2 := D_2^{-1}(B^T D_1^{-1} B x_2 + u)$

Multilevel Methods

- The multilevel version
  - Recursive solution
  - Multilevel Schur complement approximation
  - Not really based on the fundamental multigrid principles of smoothing and coarse-level correction.

1. $x_1 := D_1^{-1} b_1$
2. If $k = Level_{max}$
3. Compute $M_S^{-1} \cong S^{-1}$
   $x_2 := M_S^{-1}(b_2 - B^T x_1)$
4. Else
5. MAMI($S,\ x_2,\ k+1$)
6. End If
7. $x_1 := x_1 - D_1^{-1} B x_2$

Iterative Refinement Scheme

- **Iterative refinement**
  - **Enhance the robustness of the $\mathcal{H}$-matrix-based approximate inverse method**
  - **Linear iteration**

$$x_0 = 0, x_{i+1} = x_i + \widetilde{A^{-1}}(e_i - Ax_i)$$

  - **Convergence rate**

$$R = \left\| I - \widetilde{A^{-1}}A \right\|$$

  - **Advantage: low extra computational cost**

Experimental Results

- Proposed algorithms
  - C++ implementation
  - HLIBpro library is adopted to perform $\mathcal{H}$-matrix construction
- Experimental platform
  - Linux Server with Intel CPU@2.33GHz and 8GB RAM
- Comparison
  - ICCG solver with IC(0) preconditioner
  - Cholmod solver from SuiteSparse package

Experimental Results

- **Comparison with ICCG and Cholmod**

Runtime (second)
Peak Memory (B)

Grid Size	$\mathcal{H}$-matrix				Cholmod			ICCG
	Setup	Solve	Memory	Avg. Error	Setup	Solve	Memory	Solve
5875	0.62	0.02	7.50M	4.9E-4	0.18	0.03	5.42M	0.02
22939	3.48	0.08	33.74M	2.5E-4	0.76	0.12	30.09M	0.13
35668	6.17	0.13	53.55M	1.2E-3	0.93	0.2	52.42M	0.23
51195	9.55	0.19	83.01M	9.7E-4	1.36	0.31	84.37M	0.36
90643	18.83	0.35	161.37M	2.5E-3	2.61	0.54	176.05M	0.78
141283	31.94	0.58	254.48M	2.0E-3	4.54	0.89	302.26M	1.60
203725	65.97	0.89	479.94M	1.2E-3	6.92	1.28	469.77M	2.73
277559	94.71	1.22	670.13M	3.4E-3	8.74	1.64	687.82M	4.82
562363	206.24	2.56	1.39G	1.1E-3	26.39	3.87	1.63G	12.07
681265	344.76	3.29	2.04G	1.0E-3	31.68	4.54	2.09G	16.48
953245	443.93	4.54	2.72G	9.9E-4	45.57	6.38	3.08G	32.87
1446655	802.83	7.16	4.60G	4.4E-3	81.13	9.82	5.61G	87.29

Experimental Results

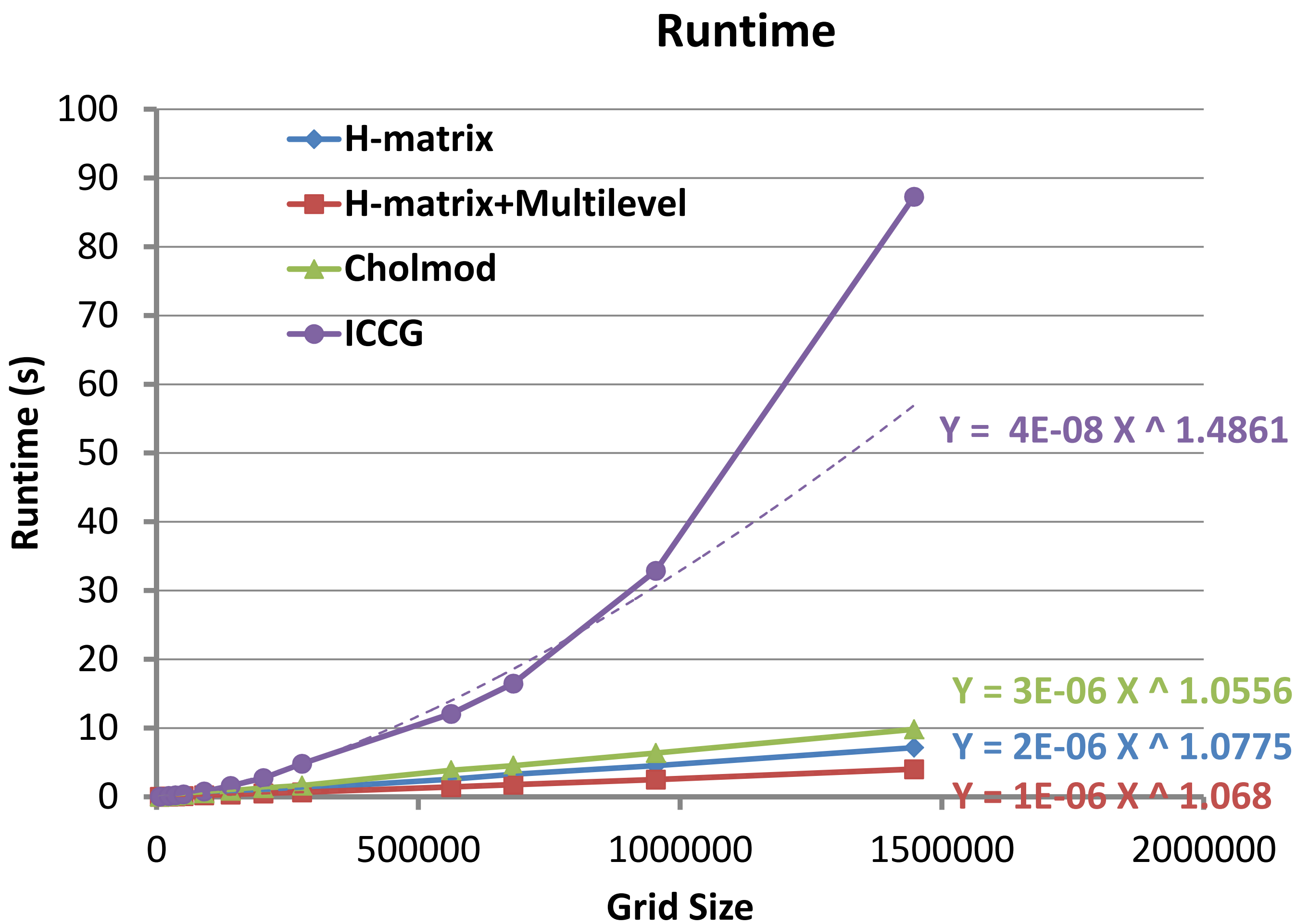
Runtime (second)
Peak Memory (B)

- With multilevel approach

Grid Size	$\mathcal{H}$-matrix				$\mathcal{H}$-matrix + Multilevel			
	Setup	Solve	Memory	Avg. Error	Setup	Solve	Memory	Avg. Error
5875	0.62	0.02	7.50MB	4.9E-4	0.46	0.01	3.80MB	1.3E-3
22939	3.48	0.08	33.74MB	2.5E-4	2.72	0.05	19.89MB	3.9E-4
35668	6.17	0.13	53.55MB	1.2E-3	4.25	0.08	29.14MB	6.6E-4
51195	9.55	0.19	83.01MB	9.7E-4	6.57	0.12	43.79MB	1.2E-3
90643	18.83	0.35	161.37M	2.5E-3	14.37	0.22	87.05MB	2.1E-3
141283	31.94	0.58	254.48M	2.0E-3	21.87	0.34	127.76M	2.7E-3
203725	65.97	0.89	479.94M	1.2E-3	37.53	0.50	196.14M	1.2E-3
277559	94.71	1.22	670.13M	3.4E-3	55.40	0.66	295.32M	2.3E-3
562363	206.24	2.56	1.39GB	1.1E-3	155.79	1.42	671.49M	1.2E-3
681265	344.76	3.29	2.04GB	1.0E-3	220.04	1.76	910.42M	1.2E-3
953245	443.93	4.54	2.72GB	9.9E-4	371.59	2.50	1.33GB	2.0E-3
1446655	802.83	7.16	4.60GB	4.4E-3	1833.54	4.01	2.45GB	4.7E-3

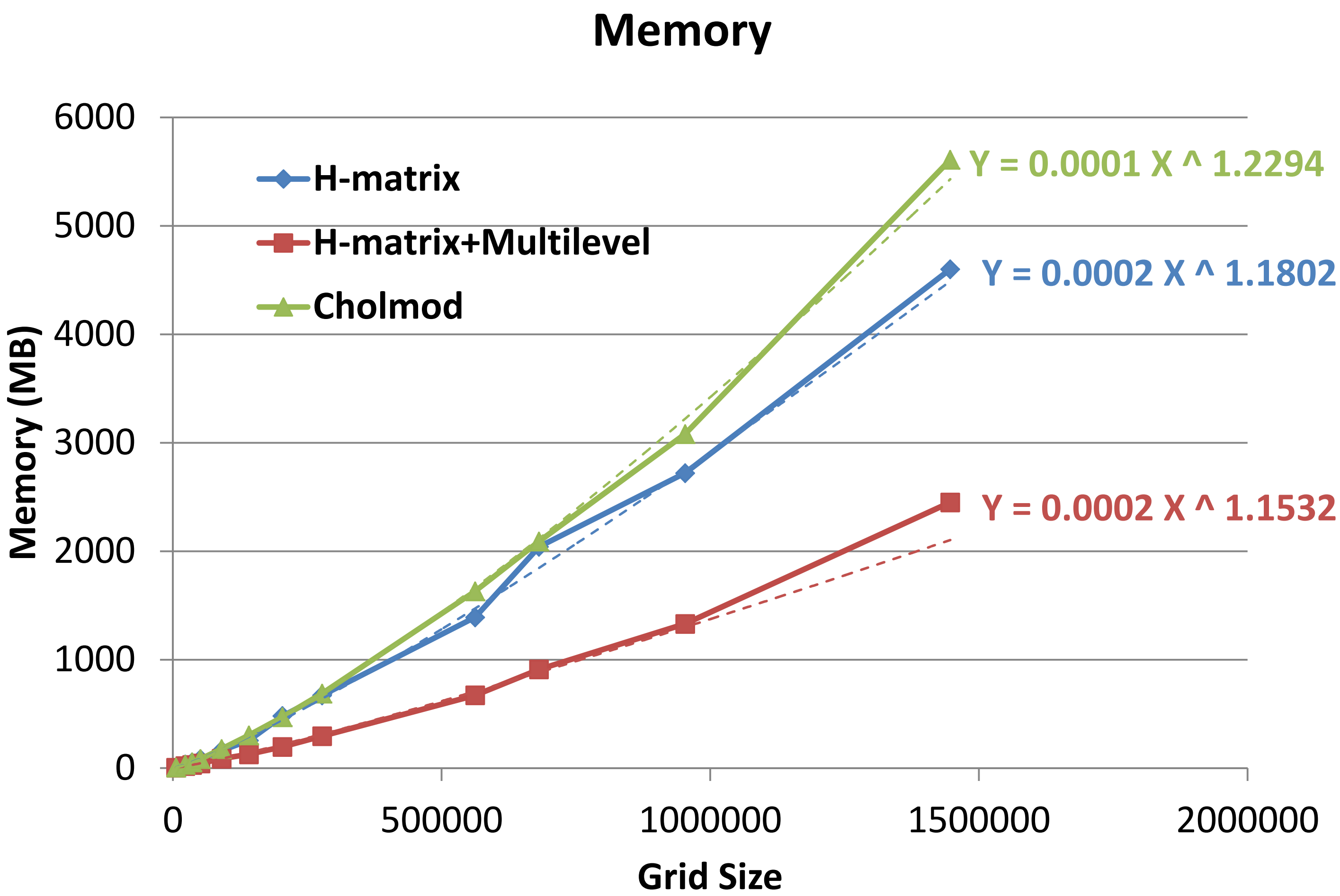
Experimental Results

- **Solve time comparison**



Experimental Results

- Memory usage comparison



Summary

- This paper proposed a multilevel $\mathcal{H}$-matrix-based approximate matrix inversion algorithm for vectorless power grid verification.
- The combination of the $\mathcal{H}$-matrix-based technique and the multilevel method is successful. And the proposed algorithm can obtain an almost linear complexity.
  - The proposed method can be also used for other occasions where linear systems with multiple right-hand sides problem needs to be solved.

THANKS FOR YOUR ATTENTION!

Q & A