A Practical Randomized GMRES Algorithm for Solving Linear Equation System In Circuit Simulation

Baiyu Chen, Jiawen Cheng, Wenjian Yu*

Department of Computer Science and Technology

Tsinghua University

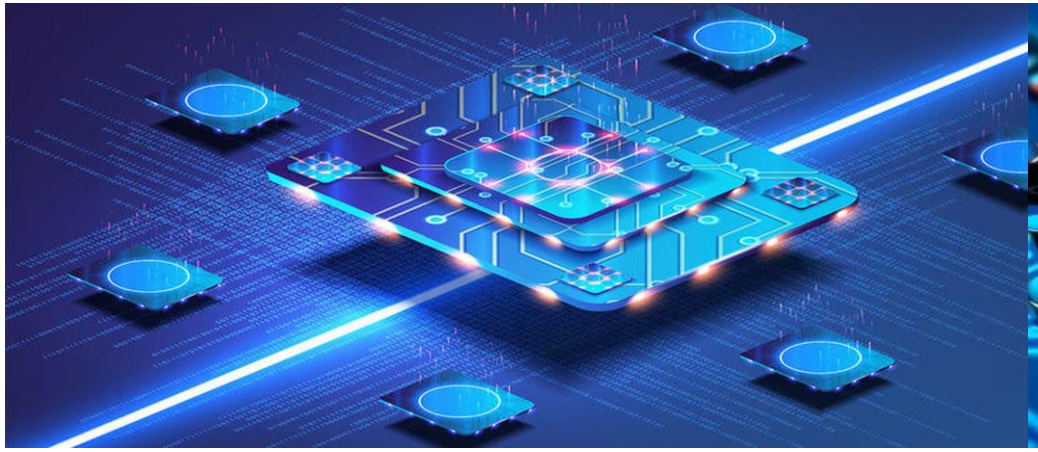
Presenter: BAIYU CHEN

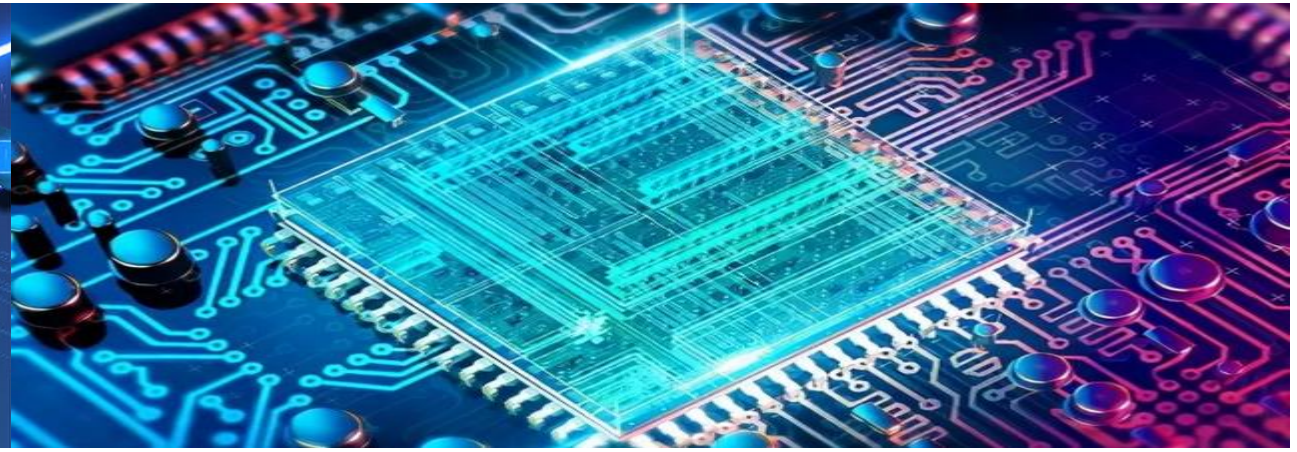
Contents

Contents

Circuit Simulation

- With the advance of chip technology, circuits with billions or even more nodes need to be simulated efficiently and effectively.
- Simulation for large-scale integrated circuits is of significance.





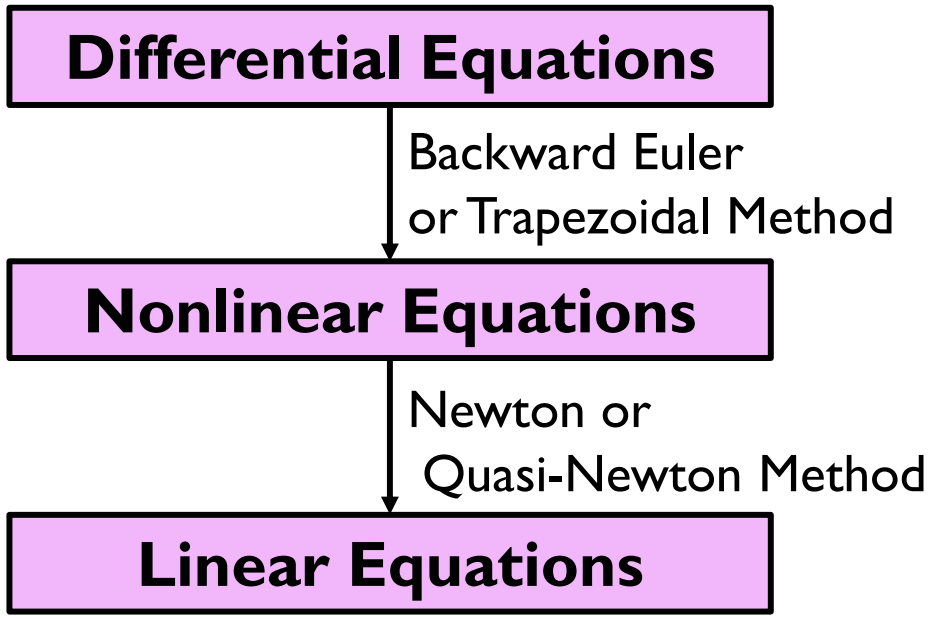
Circuit Simulation

- A typical framework:

(1) With numerical methods, transform **differential equations** into **non-linear equations**.

(2) With Newton method, transform **non-linear equations** to **linear equations**.

(3) Solve the constructed **linear equations**. (**the key process**)



Circuit Simulation

- The two key properties of linear equations from circuit simulation problem:

(1) **Sparse**: while the nodes of circuits are enormous, their connections are very sparse.

Therefore, the coefficient matrices are sparse.

(2) **Unsymmetric**: the dissymmetry can be caused by many factors. The increasing dissymmetry can be observed if the integrated chips grow larger.

- We should leverage the above two properties for more efficient solution.

Solution for Linear Equations

- The two typical types of methods for solving linear equations:

(1) Direct methods: decompose the matrix and solve by substitution.

Two main weaknesses: because of the fill-ins in the decomposition process and we cannot control the accuracy of the solution.

(2) Iterative methods: transform the solution process into many-times matrix-vector multiplication (**keep the sparsity**), and **return the solution when the demanded accuracy is met**.

CG VS GMRES

- The two most commonly used iterative methods for solving linear equations:

(1) conjugate gradient (**CG**) method: O(n) time complexity per iteration, but only applicable for symmetric positive definite (SPD) matrix.

Only **235 cases** in the **2893 cases** in SuiteSparse Matrix Collection are SPD.

0 case after 2015 in SuiteSparse Matrix Collection is SPD.

(1) generalized minimal residual (**GMRES**) method: **applicable for general linear equations**.

GMRES

- **The incredible popularity**: Since the birth of GMRES, it has been cited more than 15000 times (**the most cited numerical algorithm in our best knowledge**).
- **The main shortcoming: increasing computational cost with iterations**, caused by its core: **the Arnoldi process**.

Algorithm 1 The Arnoldi Process with Modified Gram-Schmidt

Input: $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, $x_0 \in \mathbb{R}^n$, m.
Output: $V_{m+1} \in \mathbb{R}^{n\times(m+1)}$, $\overline{H}_m \in \mathbb{R}^{(m+1)\times m}$.

1: Calculate residual $r \leftarrow b - Ax_0$
2: $v_1 \leftarrow \frac{r}{\|r\|}$
3: **for** $j = 1, 2, \cdots, m$ **do**
4: $\quad w_j \leftarrow Av_j$
5: $\quad$ **for** $i = 1, \cdots, j$ **do** ▷ modified Gram-Schmidt (MGS)
6: $\quad\quad h_{i,j} \leftarrow w_j^T v_i$
7: $\quad\quad w_j \leftarrow w_j - h_{i,j}v_i$
8: $\quad$ **end for**
9: $\quad h_{j+1,j} \leftarrow \|w_j\|$
10: $\quad v_{j+1} \leftarrow \frac{w_j}{h_{j+1,j}}$
11: **end for**
12: **Return** $V_{m+1} = [v_1, v_2, \cdots, v_{m+1}]$, $\overline{H}_m = (h_{i,j})_{(m+1)\times m}$.

GMRES

- The efficient and effective of **orthogonalization** is of significance: the orthogonalization is the rooted reason for the high complexity of GMRES and its effectiveness directly influence its convergency.
- The efficient and effective **relative error estimation** is of significance: we should know the error to know when to return the solution.

Algorithm 2 The Restarted GMRES Algorithm

Input: $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, $x_0 \in \mathbb{R}^n$, $m_{\max}$, tol .
Output: $x_{appro} \in \mathbb{R}^n$.

$\gamma \leftarrow +\infty$
while $|\gamma| > tol$ **do**
  $r \leftarrow b - Ax_0, \quad g \leftarrow \|r\|, \quad v_1 \leftarrow \frac{r}{g}$
  for $j = 1, 2, \cdots, m_{\max}$ **do**
    $w_j \leftarrow Av_j$, make g a vector $\in \mathbb{R}^{j+1}$ by appending a zero
    Run the MGS process to obtain $h_{1:j,j}$ and update w_j
    $h_{j+1,j} \leftarrow \|w_j\|$ ▷ $h_{i,j}$ denotes element of matrix $\overline{H}_m$
    $v_{j+1} \leftarrow \frac{w_j}{h_{j+1,j}}$
    Apply the Givens transformations $G_1, \cdots, G_{j-1}$ to $h_{:,j}$
    Compute the Givens matrix G_j for $h_{:,j}$ to make $h_{j+1,j} = 0$
    Apply the Givens transformation G_j to g
    $\gamma \leftarrow g_{j+1}$ ▷ last element of vector g
    if $|\gamma| \le tol$ **then**
      Break
    end if
  end for
  $m \leftarrow j$
  Solve y_m with the transformed matrix $\overline{H}_m$ and g
  $x_0 \leftarrow x_0 + V_m y_m$, with $V_m = [v_1, \cdots, v_m]$
end while
Return $x_{appro} = x_0$.

Randomized GMRES

- The theoretical analysis shows that randomized Gram-Schmidt process can promote effectiveness of Arnoldi process.
- However, **it is not practical**, for the following three oblivious drawbacks:

(1) High computational cost of sketching.

(2) Low efficiency of residual estimation.

(3) Low efficiency of solving sketched LS.

Algorithm 3 The Randomized Arnoldi Process [3]

Input: $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, $x_0 \in \mathbb{R}^n$, m, random sketching $\Theta \in \mathbb{R}^{k\times n}$.
Output: $V_{m+1} \in \mathbb{R}^{n\times(m+1)}$, $\overline{H}_m \in \mathbb{R}^{(m+1)\times m}$.

1: Calculate residual $r \leftarrow b - Ax_0$
2: Initialize $\overline{H}$ to be an $(m+1)\times m$ zero matrix
3: $s_1 \leftarrow \frac{\Theta r}{\|\Theta r\|}$, $v_1 \leftarrow \frac{r}{\|\Theta r\|}$ ▷ normalize s_1
4: **for** $j = 1, 2, \cdots, m$ **do**
5: $w_j \leftarrow Av_j$
6: $p_j \leftarrow \Theta w_j$ ▷ random sketching
7: Solve $z_j = \arg\min_z \|S_j z - p_j\|$, with $S_j = [s_1, \cdots, s_j]$
8: $v_{j+1} \leftarrow w_j - V_j z_j$, with $V_j = [v_1, \cdots, v_j]$
9: $s_{j+1} \leftarrow \Theta v_{j+1}$ ▷ random sketching
10: $\overline{H}_{1:j+1,j} \leftarrow [z_j^T, \|s_{j+1}\|]^T$
11: $v_{j+1} \leftarrow \frac{v_{j+1}}{\|s_{j+1}\|}$, $s_{j+1} \leftarrow \frac{s_{j+1}}{\|s_{j+1}\|}$ ▷ normalize s_{j+1}
12: **end for**
13: **Return** $V_{m+1} = [v_1, v_2, \cdots, v_{m+1}]$, $\overline{H}_m = \overline{H}$.

The Aim of This Work

- We aim to develop a practical and more efficient randomized GMRES algorithm, **and can replace GMRES algorithm at least in circuit simulation.**
- The contribution of this work can be summarized as follows:

(1) Propose a practical randomized Arnoldi process, which can efficiently and effectively estimate residual in each iteration.

(2) Propose a linear-time complexity sketching algorithm.

(3) Develop a practical randomized GMRES (PRGMRES) solvers.

(4) Theoretical analysis of PRGMRES algorithm.

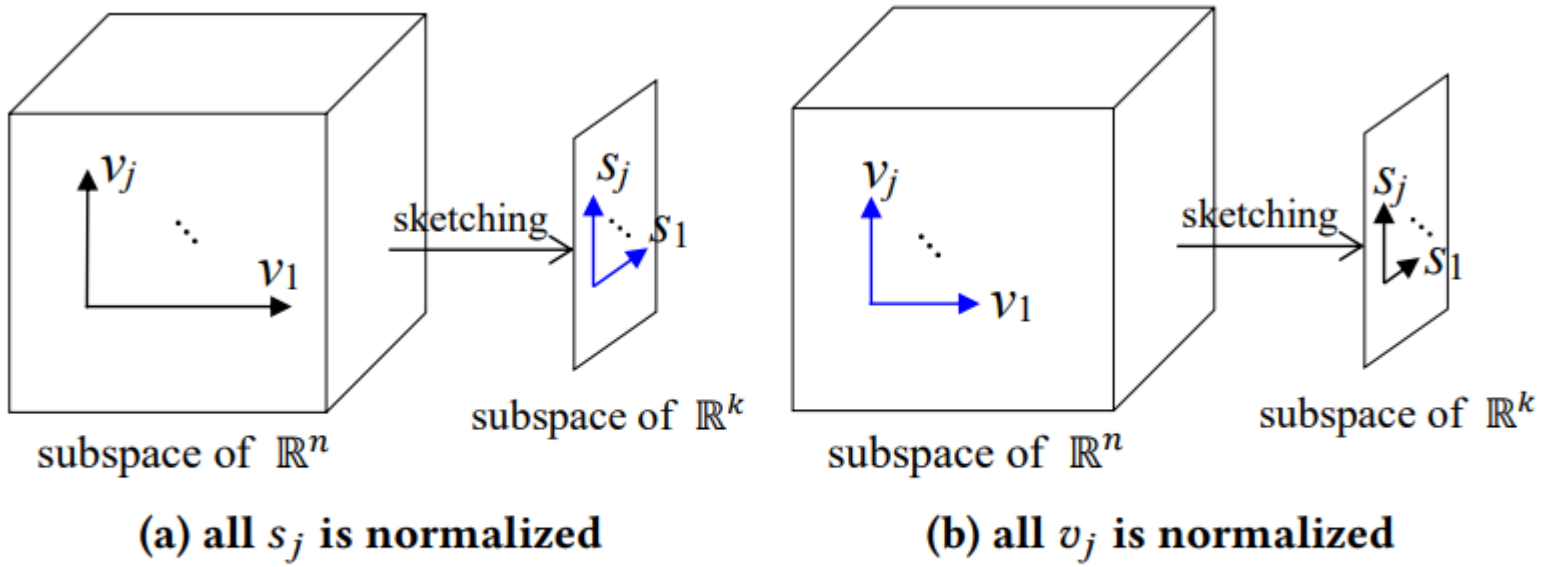
Contents

The Practical Randomized Arnoldi Process

$$\begin{aligned}\|r_m\| = \|b - Ax_m\| &= \|r_0 - AV_m y_m\| = \|r_0 - V_{m+1}\overline{H}_m y_m\| \\ &= \|V_{m+1}(\beta e_1 - \overline{H}_m y_m)\| = \|\beta e_1 - \overline{H}_m y_m\| \quad (2) \\ &= |\gamma|\,,\end{aligned}$$

- The norm of S and V is not equal with high probability due to the low rank of the sketching matrices. Therefore, the (2) does not hold and original residual estimation does not work for randomized GMRES in [3].
- To make the efficient residual estimation, we should make (2) work for randomized GMRES.
- **The key: Enforce orthonormality of V.**

The Practical Randomized Arnoldi Process



(a) all s_j is normalized **(b) all v_j is normalized**

Figure 1: Two different sketching schemes in (a) the RGMRES algorithm in [3] and (b) the proposed algorithm. Notice that the two sets of orthogonal bases satisfy $s_i = \Theta v_i$, $i = 1, 2, \cdots$.

The Practical Randomized Arnoldi Process

$$\begin{aligned}\|r_m\| = \|b - Ax_m\| &= \|r_0 - AV_m y_m\| = \|r_0 - V_{m+1}\overline{H}_m y_m\| \\ &= \|V_{m+1}(\beta e_1 - \overline{H}_m y_m)\| = \|\beta e_1 - \overline{H}_m y_m\| \quad (2) \\ &= |\gamma|\,,\end{aligned}$$

- The sketching process cannot guarantee the fully orthogonality of V. Therefore, the residual estimation is still not accurate in some extreme cases.
- **The key: implement an inner tolerance (double tolerance)**, which is smaller than demanded tolerance to avoid redundant restarting.

The Linear-Time Sketching

- The Sketching: compute $y = \tilde{\Theta}x$
- The theoretical bound of Rademacher sketching is better than P-SRHT, while the time complexity of Rademacher sketching is O(nk) and the time complexity of P-SRHT is O(nlogn) or O(nlogk).
- We aim to implement **sketching in linear complexity** by leveraging special structure.

The Linear-Time Sketching: Implicit Generation

Algorithm 4 Generate the Matrices for Linear-time Sketching

Input: n, k, ξ, ζ.

Output: $D \in \mathbb{N}^{k\times u}$, $E \in \mathbb{N}^{k\times u}$, $F \in \mathbb{N}^{k\times u}$, $P \in \mathbb{R}^{\xi\times n}$.

1: $u \leftarrow \lfloor \zeta \frac{n}{k} \rfloor$

2: Generate random matrix $P \in \mathbb{R}^{\xi\times n}$ whose element is $\frac{1}{\sqrt{k}}$ with probability 0.5 and $-\frac{1}{\sqrt{k}}$ with probability 0.5

3: **for** $i = 1, 2, \cdots, k$ **do**

4: Partition integer array $1 : n$ into u segments

5: **for** $j = 1, 2, \cdots u$ **do**

6: $d_{i,j} \leftarrow$ the left endpoint of j-th segment

7: $e_{i,j} \leftarrow$ the right endpoint of j-th segment

8: $f_{i,j} \leftarrow$ a uniformly random integer between 1 and ξ.

9: **end for**

10: **end for**

11: **Return** $D = (d_{i,j})_{k\times u}$, $E = (e_{i,j})_{k\times u}$, $F = (f_{i,j})_{k\times u}$, P.

The Linear-Time Sketching: Sketching in Linear-Time

Algorithm 5 Perform Random Sketching in Linear Time

Input: $x \in \mathbb{R}^n$, $D \in \mathbb{Z}^{k \times u}$, $E \in \mathbb{Z}^{k \times u}$, $F \in \mathbb{Z}^{k \times u}$, $P \in \mathbb{R}^{\xi \times n}$, ζ.

Output: $y \in \mathbb{R}^k$. ▷ compute $y = \tilde{\Theta}x$

1: **for** $i = 1, 2, \cdots, \xi$ **do**
2: **for** $j = 1, 2, \cdots, n$ **do**
3: Compute $t(i, j)$ with (6)
4: **end for**
5: **end for**
6: **for** $i = 1, 2, \cdots, k$ **do**
7: Compute y_i with (7), where $u = \lfloor \zeta \frac{n}{k} \rfloor$
8: **end for**
9: **Return** $y = [y_1, y_2, \cdots, y_k]^T$.

Other Skills for PRGMRES

- **Incremental Solution of the sketched least squares problem** in randomized Arnoldi process **with Householder transforms**.
- The time complexity of solving sketched least-squares problems: $O(kj^2)$ to $O(kj)$.

$$(\Theta V_j)z \approx \Theta w_j, \quad \text{i.e.} \quad \min_z \|(\Theta V_j)z - \Theta w_j\|,$$

9: Solve $z_j = \arg\min_z \|S_j z - p_j\|$, where $S_j = [s_1, \cdots, s_j]$

The Overall Algorithm

- With the above techniques, we can obtain the overall algorithm for PRGMRES algorithm.

Algorithm 6 The Practical Randomized GMRES Algorithm

Input: $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, $x_0 \in \mathbb{R}^n$, $m_{\max}$, tol, α, k, ξ, ζ .
Output: $x_{appro} \in \mathbb{R}^n$.

Generate matrices D, E, F and P with Alg. 4.
$r \leftarrow b - Ax_0, \quad g \leftarrow \|r\|$
$v_1 \leftarrow \frac{r}{g}$ ▷ normalize v_1
$s_1 \leftarrow$ sketching of v_1 with Alg. 5 ▷ linear-time sketching
while $\|r\| > tol$ **do** ▷ check the accurate residual
  for $j = 1, 2, \cdots, m_{\max}$ **do**
    $w_j \leftarrow Av_j$, make g a vector $\in \mathbb{R}^{j+1}$ by appending a zero
    $p_j \leftarrow$ sketching of w_j with Alg. 5
    Solve $z_j = \arg\min_z \|S_j z - p_j\|$, where $S_j = [s_1, \cdots, s_j]$
    $v_{j+1} \leftarrow w_j - V_j z_j$, with $V_j = [v_1, \cdots, v_j]$
    $s_{j+1} \leftarrow$ sketching of v_{j+1} with Alg. 5
    $h_{1:j+1,j} \leftarrow [z_j^T, \|v_{j+1}\|]^T$
    $v_{j+1} \leftarrow \frac{v_{j+1}}{\|v_{j+1}\|}, \quad s_{j+1} \leftarrow \frac{s_{j+1}}{\|v_{j+1}\|}$ ▷ normalize v_{j+1}
    Apply the Givens transformations $G_1, \cdots, G_{j-1}$ to $h_{:,j}$
    Compute the Givens matrix G_j for $h_{:,j}$ to make $h_{j+1,j} = 0$
    Apply the Givens transformation G_j to g
    $\gamma \leftarrow g_{j+1}$ ▷ last element of vector g
    if $|\gamma| \le \alpha \cdot tol$ **then** ▷ a slightly tighter criterion
      Break
    end if
  end for
  $m \leftarrow j$
  Obtain y_m with the transformed matrix $\overline{H}_m$ and g
  $x_0 \leftarrow x_0 + V_m y_m$, with $V_m = [v_1, \cdots, v_m]$
  $r \leftarrow b - Ax_0$ ▷ an extra computation of residual
  $g \leftarrow \|r\|, \quad v_1 \leftarrow \frac{r}{g}$
end while
Return $x_{appro} = x_0$.

Theoretical Analysis

- Results on proposed sketching algorithm.

THEOREM 1. *Suppose* $\tilde{\Theta} \in \mathbb{R}^{k \times n}$ *is a random matrix generated with the above method, then the expectation*

$$\mathbb{E}[\tilde{\Theta}^T \tilde{\Theta}] = I\,, \tag{8}$$

where I *denotes the identity matrix. Moreover, every diagonal element of* $\tilde{\Theta}^T \tilde{\Theta}$ *is always 1.*

Theoretical Analysis

- Results supporting smaller sketching size can be summarized in brief.

(1) Oblivious subspace embedding for m-dimensional Krylov subspace is not worse than any m-dimensional subspace of $\mathbb{R}^n$.

(2) Oblivious subspace embedding for the first k (k is a small number) basis vectors of Krylov subspace may have good results as well.

- The smaller sketching dimensions can be just fine.

Contents

- Background
- Proposed Method
- Experimental Results
- Conclusion

Experiments Results

- Experimental setting:

(1) Machine: Intel Xenon Gold 6230R CPU@2.10GHz and 128 RAM.

(2) Preconditioning: ILU(k) (k=3)

(3) Tolerance: 1E-8 (to show the stability of our PRGMRES algorithm)

Experiments Results

Table 1: Computational results of solving sparse linear equations. The convergence tolerance of GMRES method is all set to 1E-8, the restarting parameter is $m_{\max} = 200$, and the sketching size k of RGMRES and our PRGMRES are 1500 and 500 respectively.

Case	n	nnz	GMRES		RGMRES [3]		PRGMRES (ours)			
			$iter$	T_{tot}(s)	$iter$	T_{tot}(s)	$iter$	T_{tot}(s)	Sp_1	Sp_2
ibmpg3t	1.0E6	4.2E6	202	41.9	200	33.6	198	25.7	1.6	1.3
ibmpg4t	1.2E6	4.8E6	146	29.0	150	22.6	149	18.5	1.6	1.2
ibmpg5t	1.6E6	5.7E6	645	206	670	128	701	122	1.7	1.0
thupg1	5.3E6	2.2E7	137	132	140	103	141	96.9	1.4	1.1
thupg2	9.5E6	4.1E7	140	292	150	213	143	180	1.6	1.2
thupg3	1.2E7	5.3E7	140	402	150	290	143	245	1.6	1.2
thupg4	1.6E7	6.9E7	143	583	150	386	146	317	1.8	1.2
thupg5	2.0E7	8.8E7	150	972	150	477	154	452	2.2	1.1
scircuit	1.7E5	9.6E5	642	19.2	650	15.2	664	12.1	1.6	1.3
rajat31	4.7E6	2.0E7	113	84.4	120	68.0	115	61.4	1.4	1.1
FullChip	3.0E6	2.7E7	17	7.44	**210**	**98.8**	20	9.60	0.8	**10.3**
Freescale1	3.4E6	1.9E7	275	188	280	119	284	118	1.6	1.0
nxp1	4.1E5	2.7E6	151	11.7	160	9.86	157	8.49	1.4	1.2
nlcircuit	1.6E5	7.3E5	21	0.535	30	0.636	22	0.449	1.2	1.4
Average									1.5	1.8

n and nnz are the dimension and the number of nonzeros in coefficient matrix, respectively. $iter$ and T_{tot} denote the number of iterations and runtime of GMRES iterations, respectively. Sp_1 denotes the speedup ratio of PRGMRES to GMRES, while Sp_2 denotes the speedup ratio of PRGMRES to RGMRES.

Experiments Results

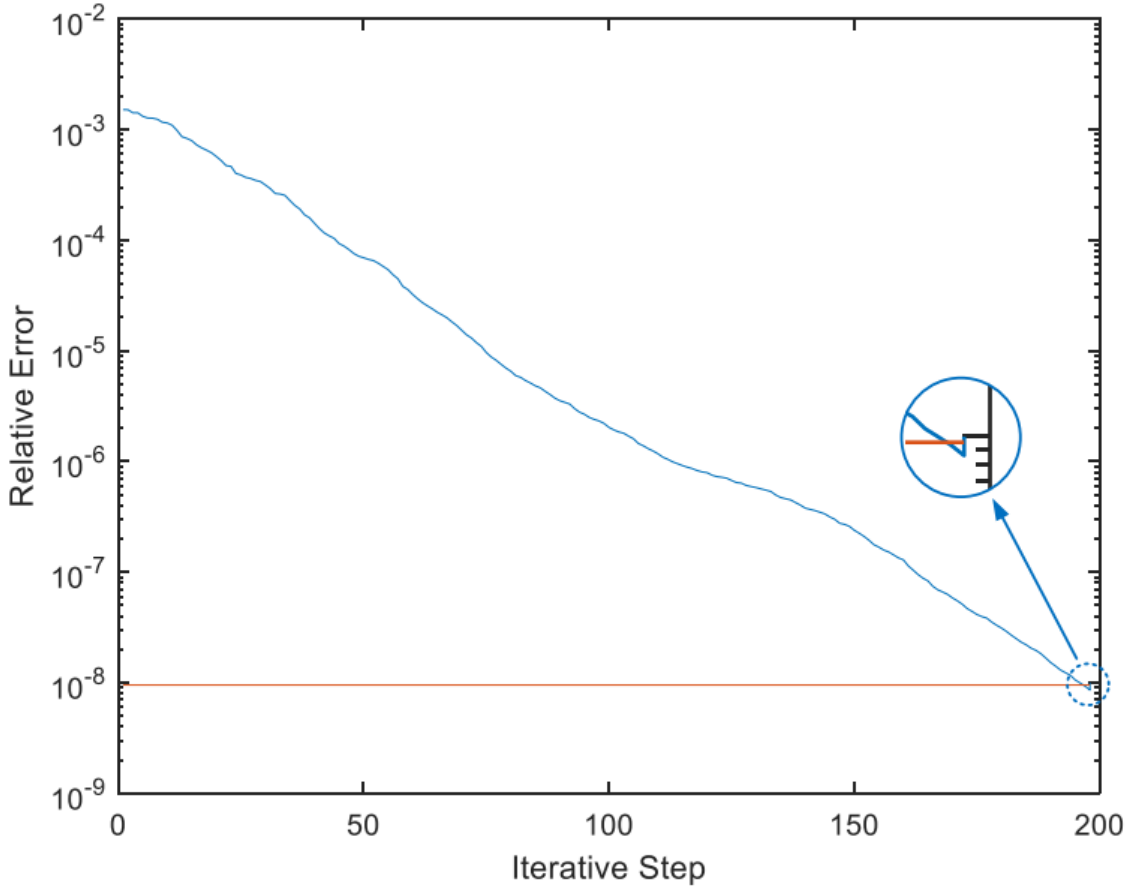


Figure 2: Convergence behavior of the relative residual error within the proposed PRGMRES algorithm for case "ibmpg3t".

Contents

- Background
- Proposed Method
- Experimental Results
- Conclusion

Conclusion

- A practical randomized Arnoldi process which can efficiently estimate residual.
- A linear-time sketching algorithm.
- The overall PRGMRES algorithm
- The theoretical analysis on PRGMRES
- The practical ILU-PRGMRES solver.

Other Possible Applications

- Below are some of other possible applications in EDA:

(1) Capacitance Extraction

(2) Multi-physics simulation

(3) Other EDA applications as long as linear equation solvers are needed

- Below are some of other possible applications besides EDA:

(1) Solving eigenvalue: proposed practical randomized Arnoldi process.

(2) Randomized singular value decomposition: proposed sketching algorithm.

Thanks

Presenter: BAIYU CHEN