A parameter tuning technique of a weighted Jacobi-type preconditioner and its application to supernova simulations

Akira IMAKURA

Center for Computational Sciences, University of Tsukuba

Joint work with

Tetsuya SAKURAI (University of Tsukuba),
Kohsuke SUMIYOSHI (Numazu College of Technology),
Hideo MATSUFURU (KEK)




Introduction

- Today's targets

>> Target algorithms

-- We consider ***preconditioning techniques*** for Krylov subspace methods to solve very large, but sparse, linear systems:

$$Ax = b, \quad A \in \mathbb{R}^{n \times n}, \quad x, b \in \mathbb{R}^n,$$

where *A* is assumed to be nonsymmetric and nonsingular.

>> Target situation

-- The algorithms are used for recent large scale simulations

-> ***high parallel efficiency*** is recognized as extremely important more than high speed and/or high accuracy

Introduction

- Today's targets
  - >> Target preconditioner
    - -- Iterative-type preconditioner based on ***Jacobi iteration***, which has high parallel efficiency

- Today's goals
  - >> ***Introduce a weighted Jacobi-type iteration*** as an improvement of the Jacobi iteration
  - >> ***Propose a parameter tuning technique*** for the weighted Jacobi-type iteration used for preconditioners
  - >> ***Evaluate the performance*** of the parameter tuning technique on the supernova simulation




Outline

- Introduction

- Weighted Jacobi-type iteration used for preconditioners
  >> Krylov subspace methods and preconditioners
  >> Weighted Jacobi-type iteration

- Parameter tuning technique

- Numerical experiments and results

- Conclusions



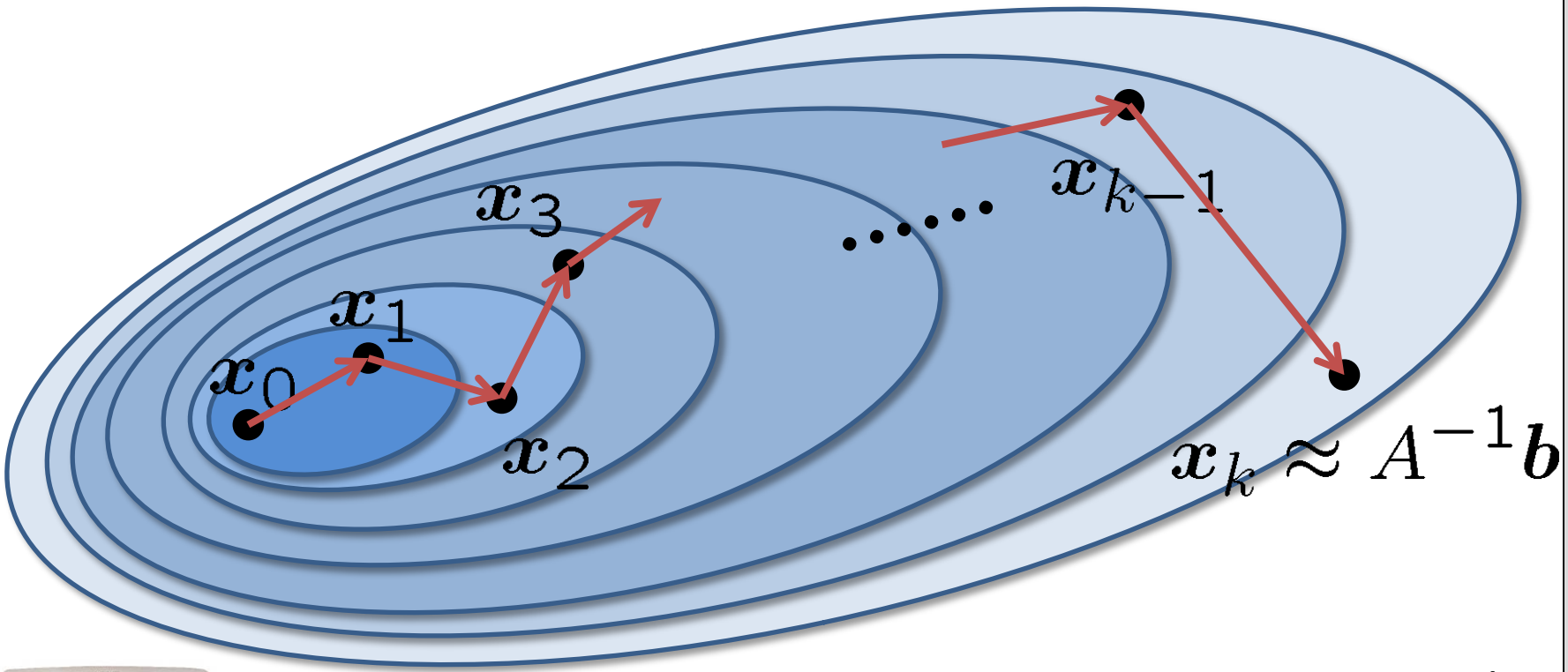

Krylov subspace methods

- Krylov subspace method

  >> Basic ideal:

  -- Projection-type iterative method based on the Krylov subspace

  >> Rough sketch (initial guess: $\boldsymbol{x}_0 \in \mathbb{R}^n$)



Preconditioners

- Preconditioners for Krylov subspace methods

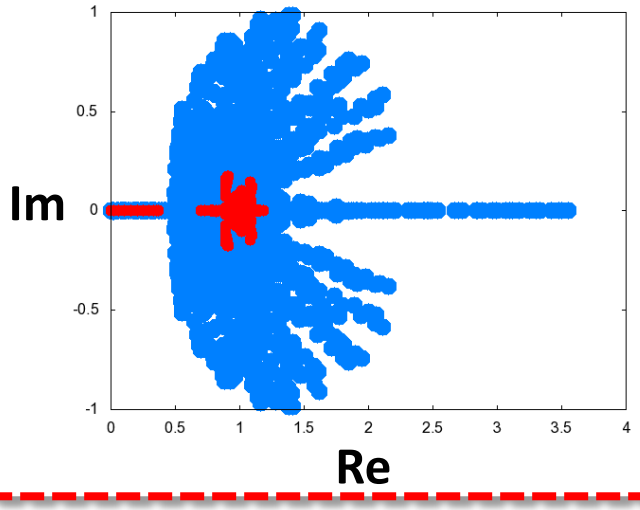
>> Convergence of Krylov subspace methods

Convergence rate depend on ***eigenvalues distribution***

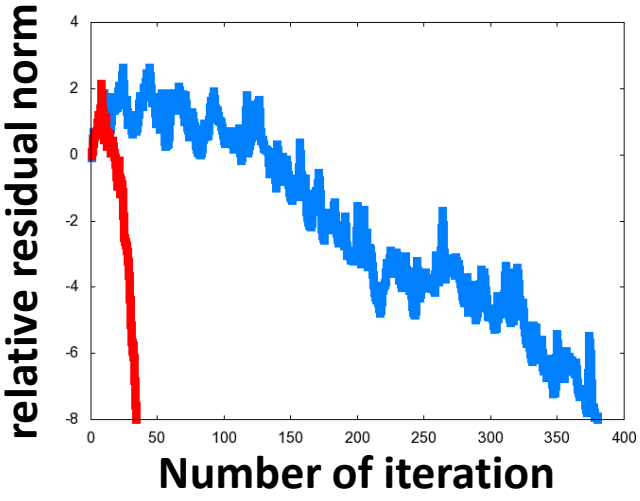
>> Preconditioners (in order to improve convergence rate)

$$Ax = b \quad \Leftrightarrow \quad AK^{-1}y = b, \quad x = K^{-1}y$$

Distribution of eigenvalues



Convergence history



Preconditioners

- How to apply preconditioners

Krylov method

$$A\,x = b$$

$$x_0 \Rightarrow x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow x_{k-1} \Rightarrow x_k$$

Computation of 1 iter. (*MatVec* + DOTs, AXPYs)

Preconditioned Krylov method

$$A\,K^{-1}\,y = b, \quad x = K^{-1}\,y$$

$$\Diamond\, y_0 \Rightarrow y_1 \Rightarrow \cdots \Rightarrow y_k, \Diamond$$

Pre-processing — Computation of 1 iter. — Post-processing

(*MatVec* + *Solve K's system* + DOTs, AXPYs)

Preconditioners

- Classification of preconditioners

>> Direct-type preconditioners

-- Construct K or K^{-1} by direct method such that $K \approx A$ and $Kw = v$ can be solved efficiently

-- In each iteration, systems $Kw = v$ is solved

Good point: small iteration cost

Bad point: large pre-processing cost, large memory

>> Iterative-type preconditioners

-- Do not construct K or K^{-1}

-- In each iteration, instead of solving $Kw = v$ $(K \approx A)$, solve roughly $Aw = v$ by some iterative method

Good point: small memory, small or no pre-processing cost

--> Efficient for recent large scale simulations

Bad point: have many parameters

Weighted Jacobi-type iteration

- Stationary iterative methods

  >> Recurrence formula ($A = M - N$)

$$\boldsymbol{x}_{k+1} = M^{-1}N\boldsymbol{x}_k + M^{-1}\boldsymbol{b}$$

  >> Convergence rate

$$\rho(G) := \rho(M^{-1}N) := \max_i \left|\lambda_i(M^{-1}N)\right|$$

$$A = -A_L - A_D - A_U$$

- Standard stationary iterative methods

	Convergence	Iteration costs	Parallelization
Jacobi	Bad	Good	Good
Gauss-Seidel	Good	Good	Bad
SOR	Bad -- Excellent	Good	Bad

Weighted Jacobi-type iteration

- **Weighted** Jacobi iteration

  >> Basic ideal:

Weighted Jacobi iteration transforms the Jacobi's recurrence formula:

$$\boldsymbol{x}_{k+1} = A_D^{-1}(A_L + A_U)\boldsymbol{x}_k + A_D^{-1}\boldsymbol{b}$$

into

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \omega\left(A_D^{-1}(A_L + A_U)\boldsymbol{x}_k + A_D^{-1}\boldsymbol{b}\right) + (1-\omega)\boldsymbol{x}_k \\ &= \boldsymbol{x}_k + \omega A_D^{-1}(\boldsymbol{b} - A\boldsymbol{x}_k) \end{aligned}$$

by using a ***weight parameter*** $\omega \in \mathbb{R}$ in order to improve the convergence rate of Jacobi iteration.

Weighted Jacobi-type iteration

- Weighted Jacobi-**type** iteration

  >> Basic ideal:

Weighted Jacobi-type iteration transforms the recurrence formula of the weighted Jacobi iteration:

$$\boldsymbol{x}_{k+1} = \omega A_D^{-1}(\boldsymbol{b} - A\boldsymbol{x}_k)$$

into

$$\boldsymbol{x}_{k+1} = \omega D^{-1}(\boldsymbol{b} - A\boldsymbol{x}_k)$$

by using some nonsingular ***scaling diagonal matrix*** D in order to improve the convergence rate without loss of its parallel efficiency.

Optimal ω and D are different with respect to each problems
--> PARAMETER TUNING TECHNIQUE IS STRONGLY REQUIRED !!

Outline

- Introduction
- Weighted Jacobi-type iteration used for preconditioners
- Parameter tuning technique
  - >> Convergence analysis
  - >> Proposal for parameter tuning technique
- Numerical experiments and results
- Conclusions




Convergence analysis

- Convergence of the weighted Jacobi-type iteration

Weighted Jacobi-type iteration can also be recognized as the stationary iterative method with the initial matrix partition:

$$A = M_\omega - N_\omega, \qquad \begin{aligned} M_\omega &= D/\omega, \\ N_\omega &= M_\omega - A \end{aligned}$$

and its convergence rate can be estimated by

$$\rho(G_\omega) := \rho(M_\omega^{-1} N_\omega) = \rho(I - \omega D^{-1} A)$$

We analyze

$$\omega^* := \arg\min_{\omega \in \mathbb{R}} \rho(G_\omega), \qquad \rho(G_{\omega^*}) := \min_{\omega \in \mathbb{R}} \rho(G_\omega)$$

Convergence analysis

- Convergence theorem of weighted Jacobi-type method

Theorem: Optimal weight parameter

Let $C(\gamma, \rho)$ ***be the inner region of the circle with center*** $\gamma \in \mathbb{R}$ ***and radius*** $\rho \in \mathbb{R}$ ***on the complex plane, and let*** $\gamma^*, \rho^* \in \mathbb{R}$ ***be defined by***

$$\gamma^*, \rho^* := \arg\min_{\gamma,\rho} \left|\frac{\rho}{\gamma}\right|,$$

$$s.t. \quad \lambda_i(D^{-1}A) \in C(\gamma, \rho), \quad i = 1, 2, \ldots, n.$$

Then we have

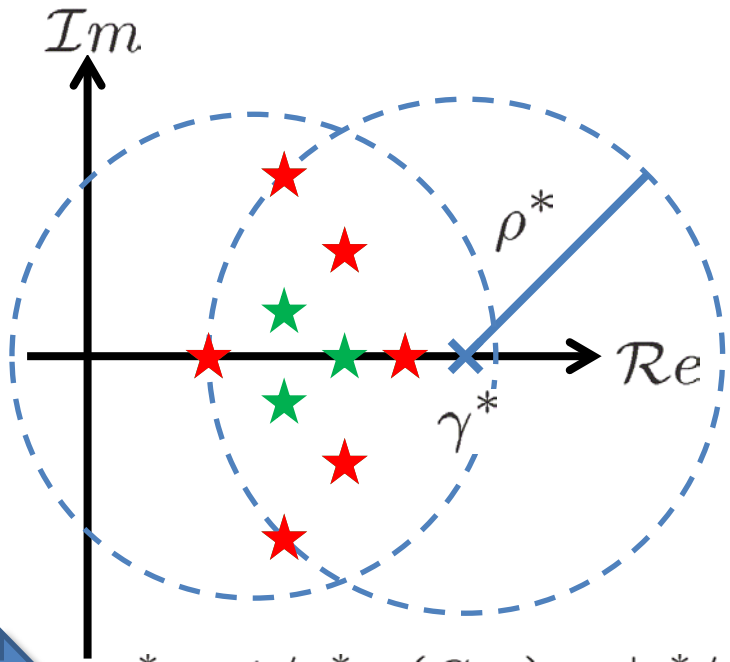
$$\omega^* := \arg\min_{\omega\in\mathbb{R}} \rho(G_\omega) = \frac{1}{\gamma^*}, \quad \rho(G_{\omega^*}) := \min_{\omega\in\mathbb{R}} \rho(G_\omega) = \left|\frac{\rho^*}{\gamma^*}\right|$$

Convergence analysis

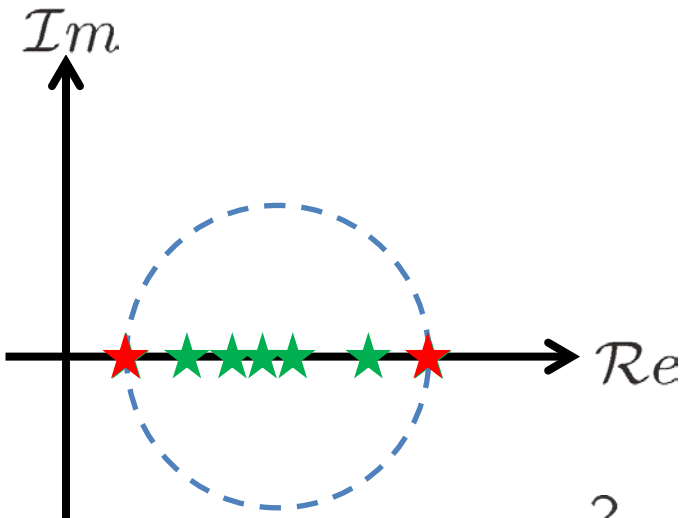
- Convergence theorem of weighted Jacobi-type method

Remark: Eigenvalue distribution and Optimal parameter

Distribution of $\lambda_i(D^{-1}A)$



$\omega^* = 1/\gamma^*, \rho(G_{\omega^*}) = |\rho^*/\gamma^*|$



$$\omega^* = \frac{2}{\lambda_{\max} + \lambda_{\min}}$$

$$\rho(G_{\omega^*}) = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}$$

Weight parameter can be optimized by ***extreme eigenvalues***

Parameter tuning technique

- Parameter tuning technique

  >> Basic ideal: ***Offline tuning***

  -- ***ω*** and ***D*** are tuned before the algorithm runs.

Input

Initial guess x_0

Choice of diagonal matrices $D_i, i = 1, \ldots, s$

Parameter tuning

1: For *i* = 1, 2, …

2: Optimize $\tilde{\omega}_i$ and get $\rho(G_{\tilde{\omega}_i})$ from approx. extreme eigenvalues $\lambda(D_i^{-1}A)$

3: End For

4: Select the best pair $(\tilde{\omega}_i, D_i)$ s.t. $\min \rho(G_{\tilde{\omega}_i})$

Preconditioned Krylov

Run weighted Jacobi-type preconditined Krylov subspace method with tuned pair $\tilde{\omega}$, D

Parameter tuning technique

- Algorithm of parameter tuning technique
  - >> Approx. eigenvalues are computed by ***the Arnoldi method***
  - >> Iteration of Arnoldi method is stopped by ***relative error of ω***

Parameter tuning

1: For i = 1, 2, …, s
2: For l = 1, 2, …, l_{max}
3: Operate lth step of the Arnoldi method and get l approx. extreme eigenvalues
4: Compute $\tilde{\omega}_l$ and $\rho(G_{\tilde{\omega}_l})$ from computed eigenvalues
5: IF $|\tilde{\omega}_l - \tilde{\omega}_{l-1}/\tilde{\omega}_l| \leq \epsilon$ then exit
6: End For
7: Set $\tilde{\omega}_i = \tilde{\omega}_l, \rho(G_{\tilde{\omega}_i}) = \rho(G_{\tilde{\omega}_l})$
8: End For
9: Select the best pair $(\tilde{\omega}_i, D_i)$ s.t. $\min \rho(G_{\tilde{\omega}_i})$

Outline

- Introduction
- Weighted Jacobi-type iteration used for preconditioners
- Parameter tuning technique
- Numerical experiments and results
  - >> Evaluation of the quality of the parameter tuning
  - >> Evaluation of the performance of the preconditioner
  - >> Evaluation of the performance for real application
- Conclusions




Numerical experiments I

- Purpose: Evaluation of quality of parameter tuning for ω

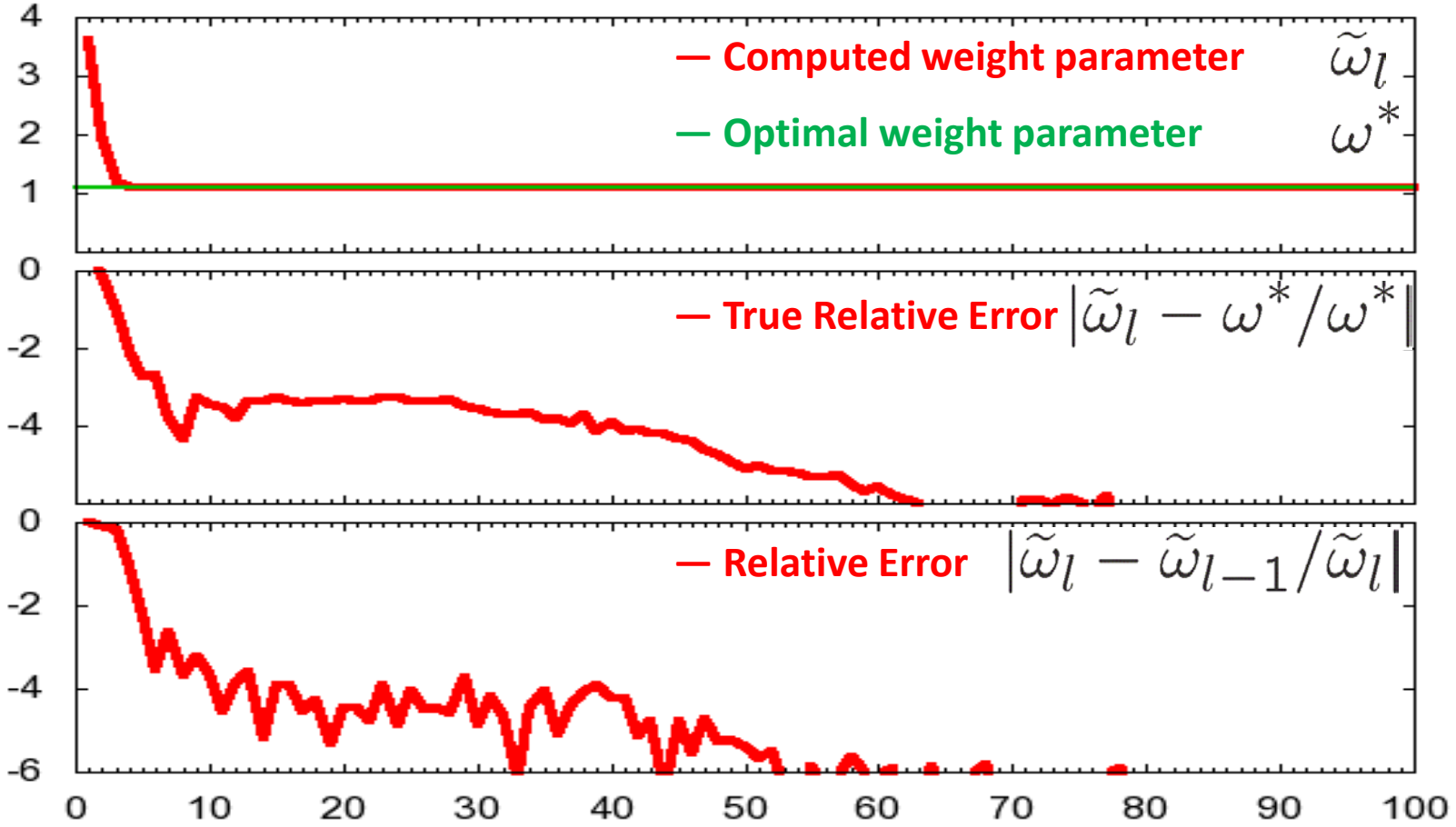
- Test matrices

>> CAVITY05	n = 1182	Fluid dynamics
>> FLOWMETER5	n = 9669	Model reduction
>> RAEFSKY2	n = 3242	Fluid dynamics
>> SHERMAN5	n = 3312	Fluid dynamics

- Experimental condition
  - >> OS: CentOS, CPU: Intel Xeon X5550 (2.67GHz), Memory: 48GB
  - >> Compiler: GNU Fortran ver.4.1.2, Compile option: -O3

Numerical results I

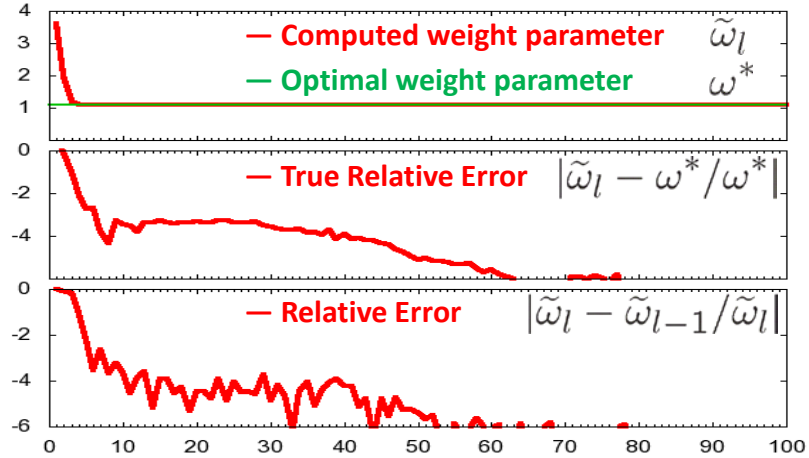
- Convergence history of ω
  - >> CAVITY05



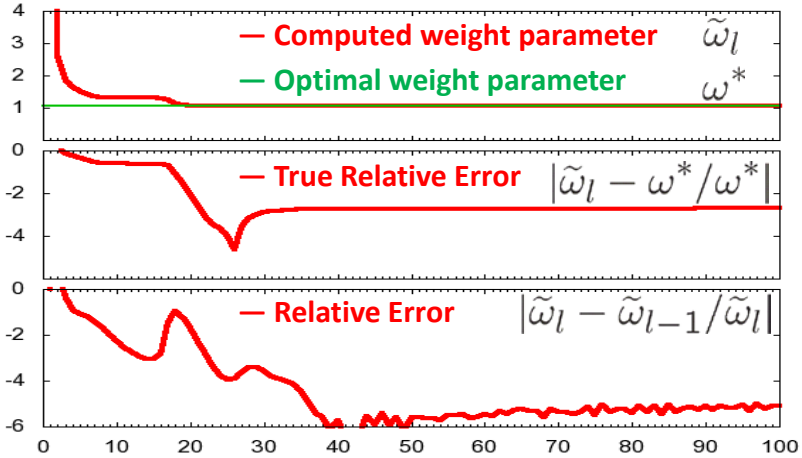
Numerical results I

- Convergence history of ω

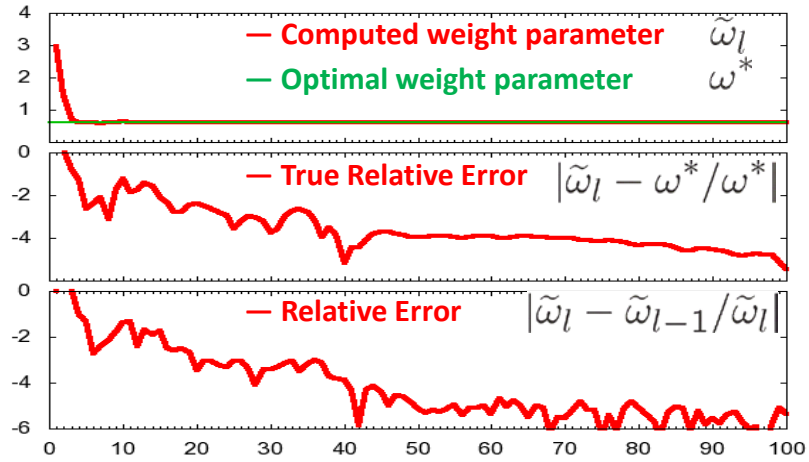
>> CAVITY05



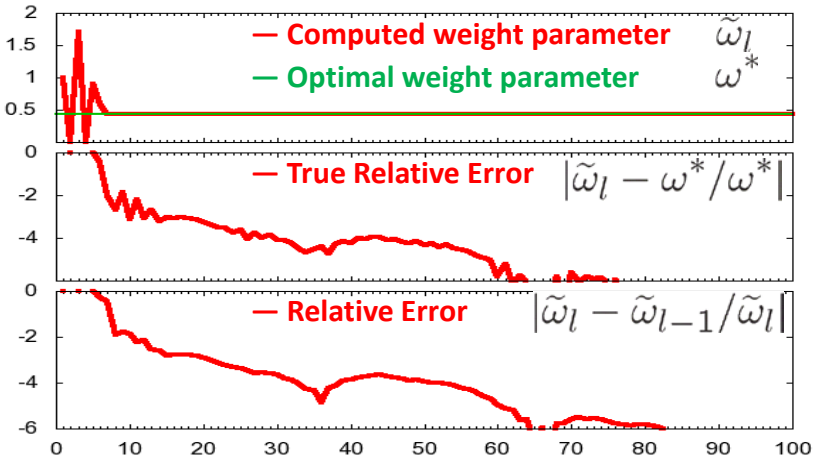
>> FLOWMETER5



>> RAEFSKY2



>> SHERMAN5



Numerical experiments II

- Purpose: Evaluation of performance of preconditioner

- Comparison
  - >> with the tuned parameters
  - >> with the default parameters (same as Jacobi iteration)

- Test matrices (in addition to matrices used in experiments I)

>> AF23560	n = 23560	Fluid dynamics
>> CHIPCOOL0	n = 20082	Model reduction
>> EPB2	n = 25228	Thermal problem
>> FEM_3D_THERMAL2	n = 147900	Thermal problem
>> POISSON3DA	n = 13514	Fluid dynamics
>> POISSON3DB	n = 85623	Fluid dynamics
>> XENON1	n = 48600	Materials problem
>> XENON2	n = 157464	Materials problem

Numerical experiments II

- Other settings
  - >> $\boldsymbol{b} = [1, 1, \ldots, 1]^{\mathsf{T}}, \boldsymbol{x}_0 = [0, 0, \ldots, 0]^{\mathsf{T}}$
  - >> Krylov subspace method: full-GMRES method
  - >> Stopping criterion for Krylov method: $\|\boldsymbol{r}_k\|_2 / \|\boldsymbol{r}_0\|_2 \leq 10^{-10}$
  - >> Stopping criterion for Arnoldi method: $|\widetilde{\omega}_l - \widetilde{\omega}_{l-1} / \widetilde{\omega}_l| \leq 10^{-4}$
  - >> Number of weighted Jacobi-type iteration: 20
  - >> Choice of scaling diagonal matrices:

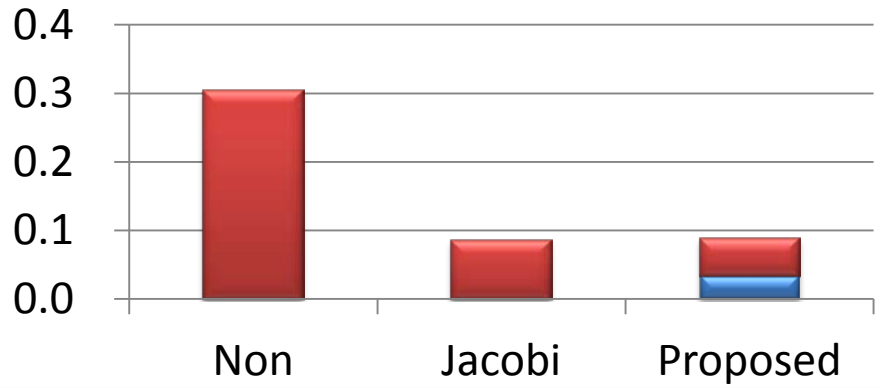
$$
\begin{aligned}
D_1 &: d_i = a_{i,i}, \\
D_2 &: d_i = 1, \\
D_3 &: d_i = \sum_{j=1}^{n} |a_{i,j}|, \\
D_4 &: d_i = \sqrt{\sum_{j=1}^{n} a_{i,j}^2}, \\
D_5 &: d_i = \max_j |a_{i,j}|, \\
D_6 &: \min_D \|D^{-1}A - I\|_F,
\end{aligned}
$$

Numerical results II

: Krylov iteration
: tuning

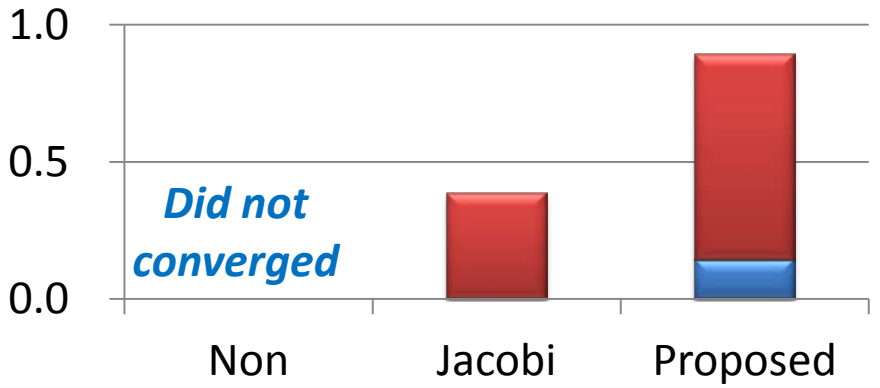
of iteration and Computation time [sec.]

>> CAVITY05



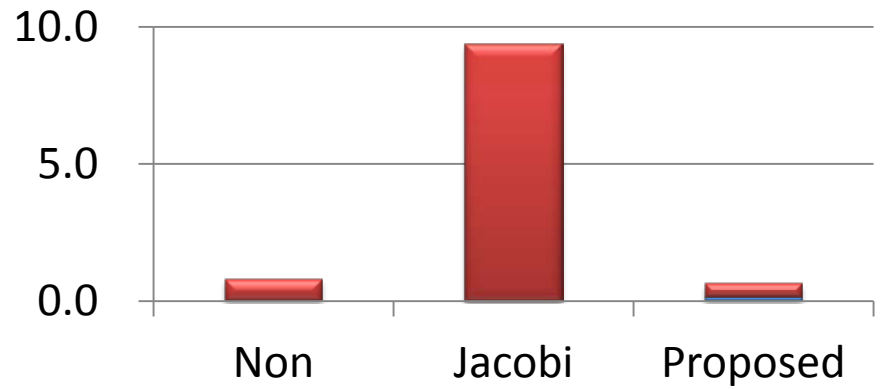
ITER	471	64	42

>> FLOWMETER5



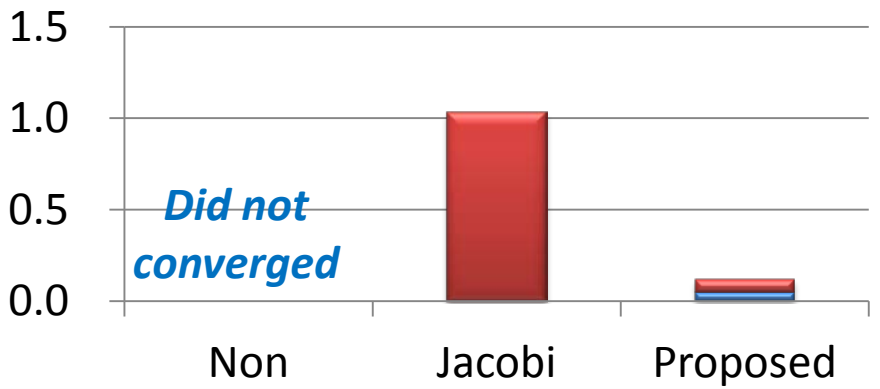
ITER	--	82	140

>> RAEFSKY2



ITER	389	750	50

>> SHERMAN5



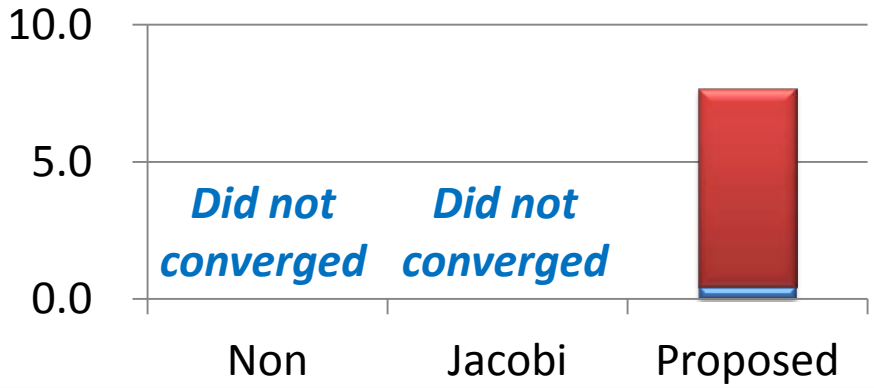
ITER	--	363	47

Numerical results II

: Krylov iteration
: tuning

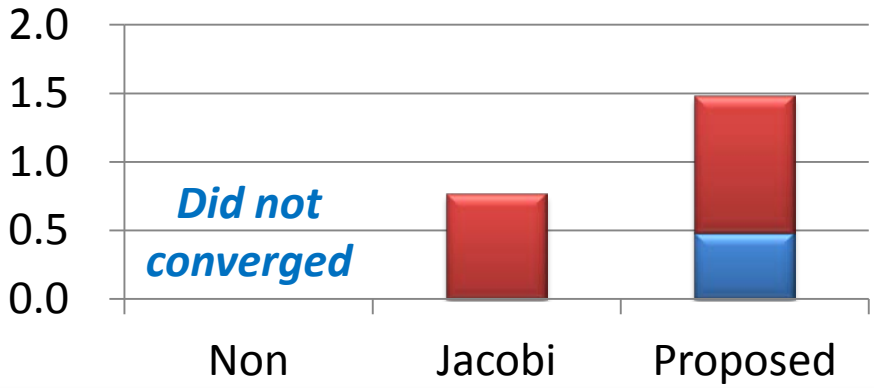
of iteration and Computation time [sec.]

>> AF23560



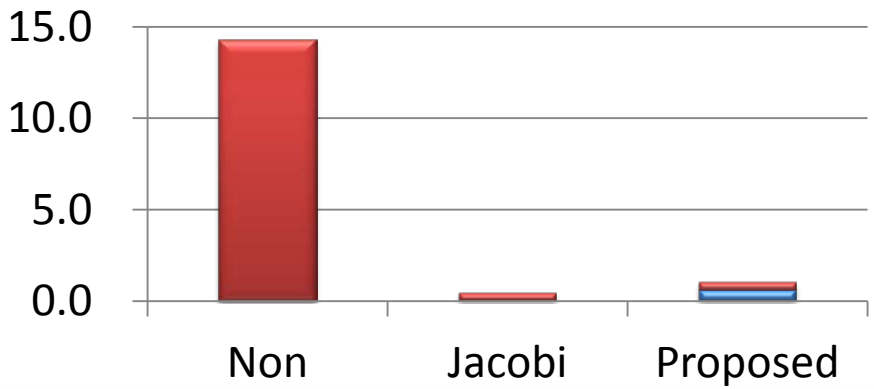
	Non	Jacobi	Proposed
ITER	--	--	259

>> CHIPCOOL0



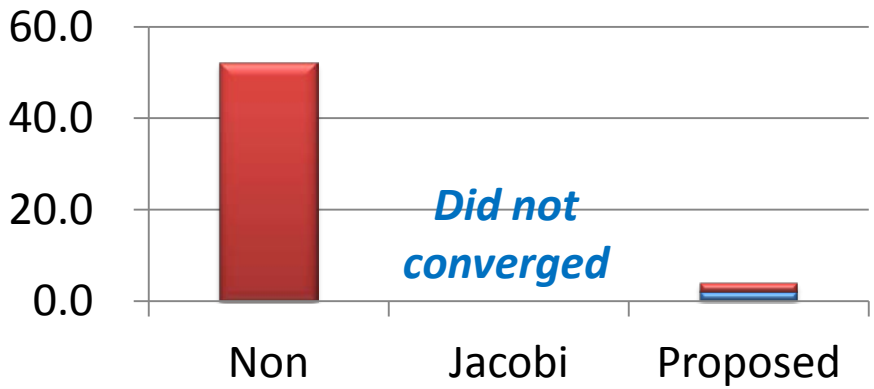
	Non	Jacobi	Proposed
ITER	--	49	63

>> EPB2



	Non	Jacobi	Proposed
ITER	650	39	36

>> FEM_3D_THERMAL2



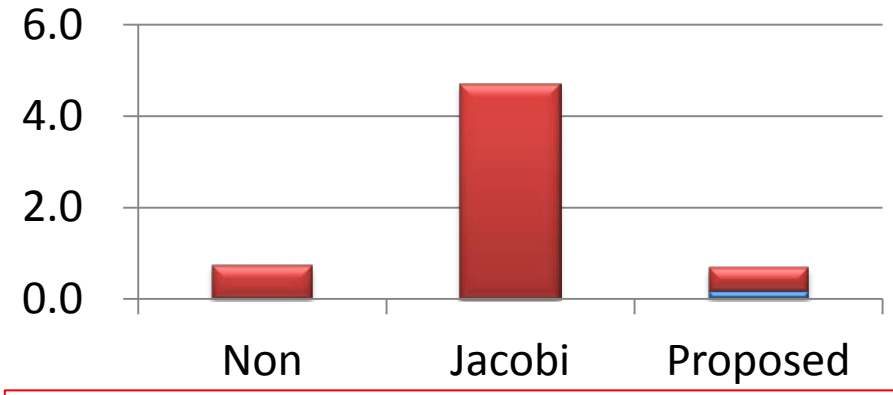
	Non	Jacobi	Proposed
ITER	492	--	11

Numerical results II

: Krylov iteration
: tuning

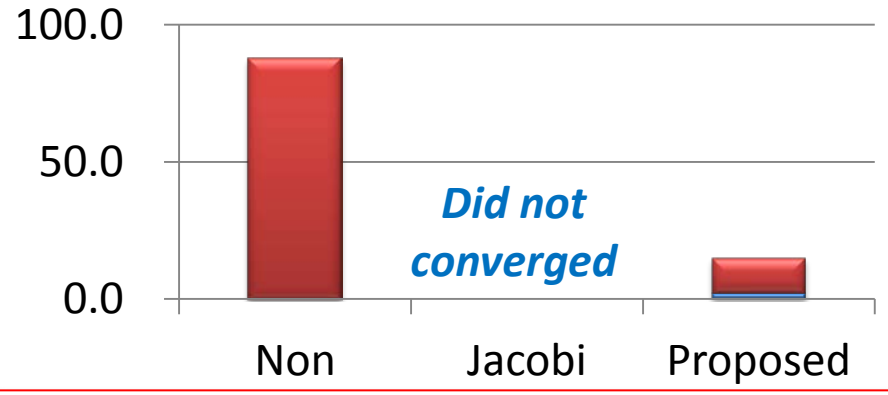
of iteration and Computation time [sec.]

>> POISSON3DA



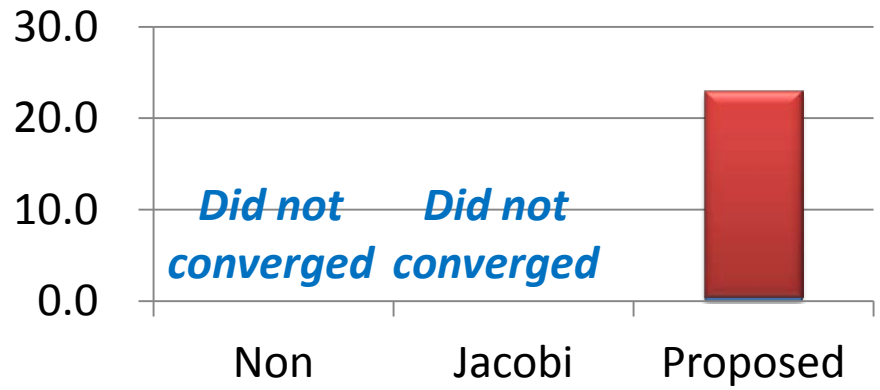
	Non	Jacobi	Proposed
ITER	184	293	31

>> POISSON3DB



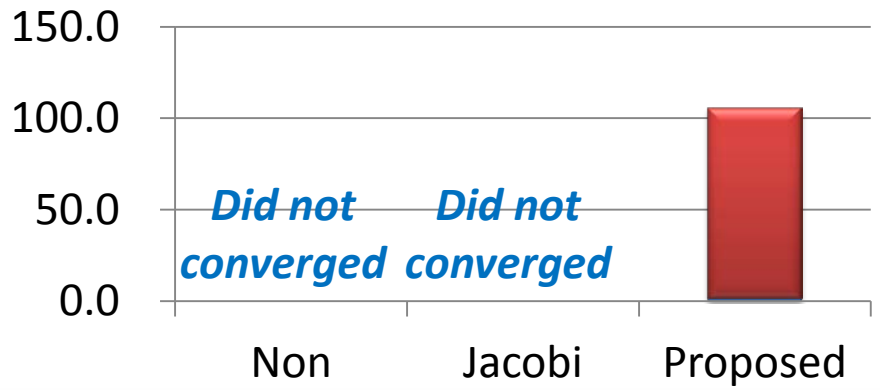
	Non	Jacobi	Proposed
ITER	848	--	66

>> XENON1



	Non	Jacobi	Proposed
ITER	--	--	284

>> FEM_3D_THERMAL2



	Non	Jacobi	Proposed
ITER	--	--	383

Numerical experiments III

- Purpose: Evaluation of performance for real application

- Target simulation
  - >> 3D supernova simulation
    - -- Time evolution : Implicit method
      - → It is required to solve ***very large, but sparse, linear systems***
        (n ≃ 16 million, Nnz ≃ 1.28 billion, Nnz/n ≃ 80)
    - -- Required time evolution : t = 1 [sec.]
      - → Larger time step (Δt) is required ($\Delta t \approx 10^{-5}$ [sec.])
    - -- ***Larger Δt leads to more ill-conditioned systems***

- Compared with
  - >> Jacobi preconditioner
  - >> Diagonal scaling

Numerical experiments III

- Other settings
  - >> Krylov subspace methods: Bi-CGSTAB
  - >> Iteration for preconditioner : 20
  - >> Cutoff parameter : 0.01
  - >> Diagonal matrix :

$$D_1 \ : \ d_i = a_{i,i},$$

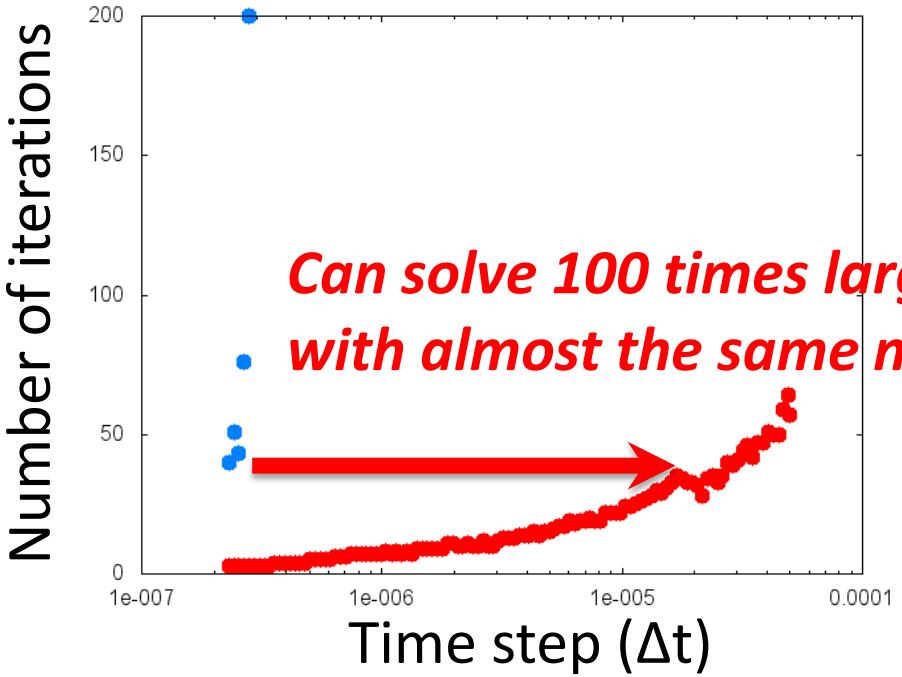
$$D_2 \ : \ d_i = \sum_{j=1}^{n} |a_{i,j}|,$$

$$D_3 \ : \ d_i = \sqrt{\sum_{j=1}^{n} a_{i,j}^2}$$

- Experimental condition
  - >> KEK SR16000/M1
    - -- 1 node 32 cores (logical 64 cores)

Numerical results III

- Numerical results
  - >> Time step (Δt) v.s. number of iterations
    - -- Bi-CGSTAB with ***Diagonal scaling*** / ***Proposed preconditioner***



  - >> Computation time
    - -- ***Diagonal scaling***: 0.3 [sec.] / iter
    - -- ***Proposed***: 3.0 [sec.] / iter (parameter tuning: 9.0 [sec.])

Outline

- Introduction
- Weighted Jacobi-type iteration used for preconditioners
- Auto-tuning technique
- Numerical experiments and results
- Conclusions




Conclusion

- Conclusion

  >> In this presentation, we have

  -- introduced ***a weighted Jacobi-type iteration***

  -- proposed ***a parameter tuning technique***

  >> From our numerical experiments, we have leaned that our tuning technique well played for solving linear systems.

- Further investigation

  >> Further improvement of the parameter tuning technique

  -- Evaluate the parallel efficiency

  -- Apply to other application, and evaluate its efficiency