An Asynchronous Distributed and Parallel Unite and Conquer Method to Solve Sequences of Non-Hermitian Linear systems

Serge G. Petiton and Xinzhe Wu

Collaboration with He Haiwu, Guy Bergère, Azedine Essai, Ye Zhang, Salim Nahi, and Pierre-Yves Aquilenti

October 23, 2018

Outline

- Introduction
- The GMRES-LS/ERAM method
- The Distributed and Parallel “Unite and Conquer GMRES/LS ERAM” method (UCGLE)
- Numerical Experiments
- Conclusion

Outline

- **Introduction**
- The GMRES-LS/ERAM method
- The Distributed and Parallel “Unite and Conquer GMRES/LS ERAM” method (UCGLE)
- Numerical Experiments
- Conclusion

Introduction

- It exists a lot of applications requiring to compute sequences of non Hermitian very sparse very large Linear Systems, such as :

$$A x_i = b_i$$

- Matrix factorization (such as LU) is not efficient for very sparse matrices (fill in of elements, really difficult to efficiently optimize)
- Deflated Krylov [Erhel 96] methods may be an option

We also want to use existing parallel libraries (PETSc, Trilinos,..)

Nahid Emad gave a talk at Mathias 2017 on “Unite and Conquer Asynchronous Methods” : iterative/restarded parallel methods asynchronously distributed on several parts of a supercomputers or between different computers, exchanging partial results to accelerate the global convergence (**avoiding global operations** along all the computing units and **intrinsically fault tolerant**)

The “Unite and Conquer GMRES/LS-ERAM” (UCGLE) method has these properties.

Outline

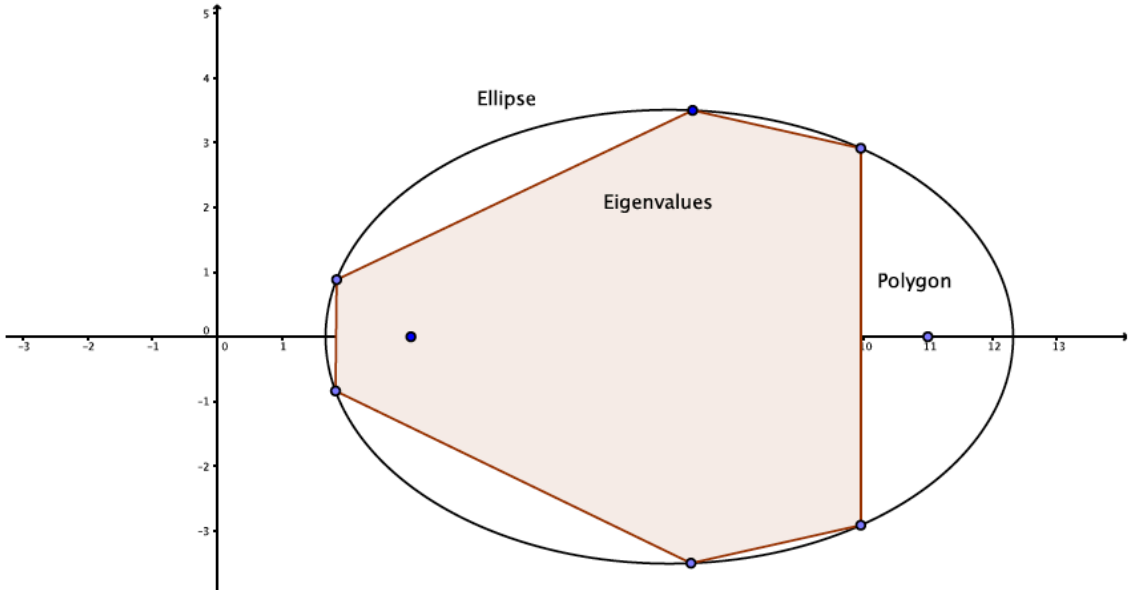
- Introduction
- **The GMRES-LS/ERAM method**
- The Distributed and Parallel “Unite and Conquer GMRES/LS ERAM” method (UCGLE)
- Numerical Experiments
- Conclusion

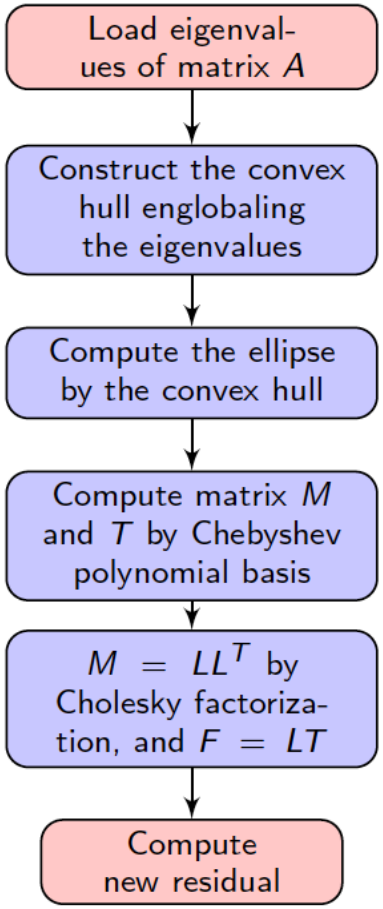
Least Squares Method

Polynomial preconditioner iterates: $x_n = x_0 + P_n(A)r_0 \rightarrow r_n = R_n(A)r_0$ with $R_n(\lambda) = 1 - \lambda P_n(\lambda)$.
The purpose is to find a kind of polynomial P_n which can minimize $R_n(A)r_0$. For more details of this method, see the article [Youssef Saad, 1987].

$$min\ max_{\lambda \in \sigma(A)} |R_d(\lambda)|$$

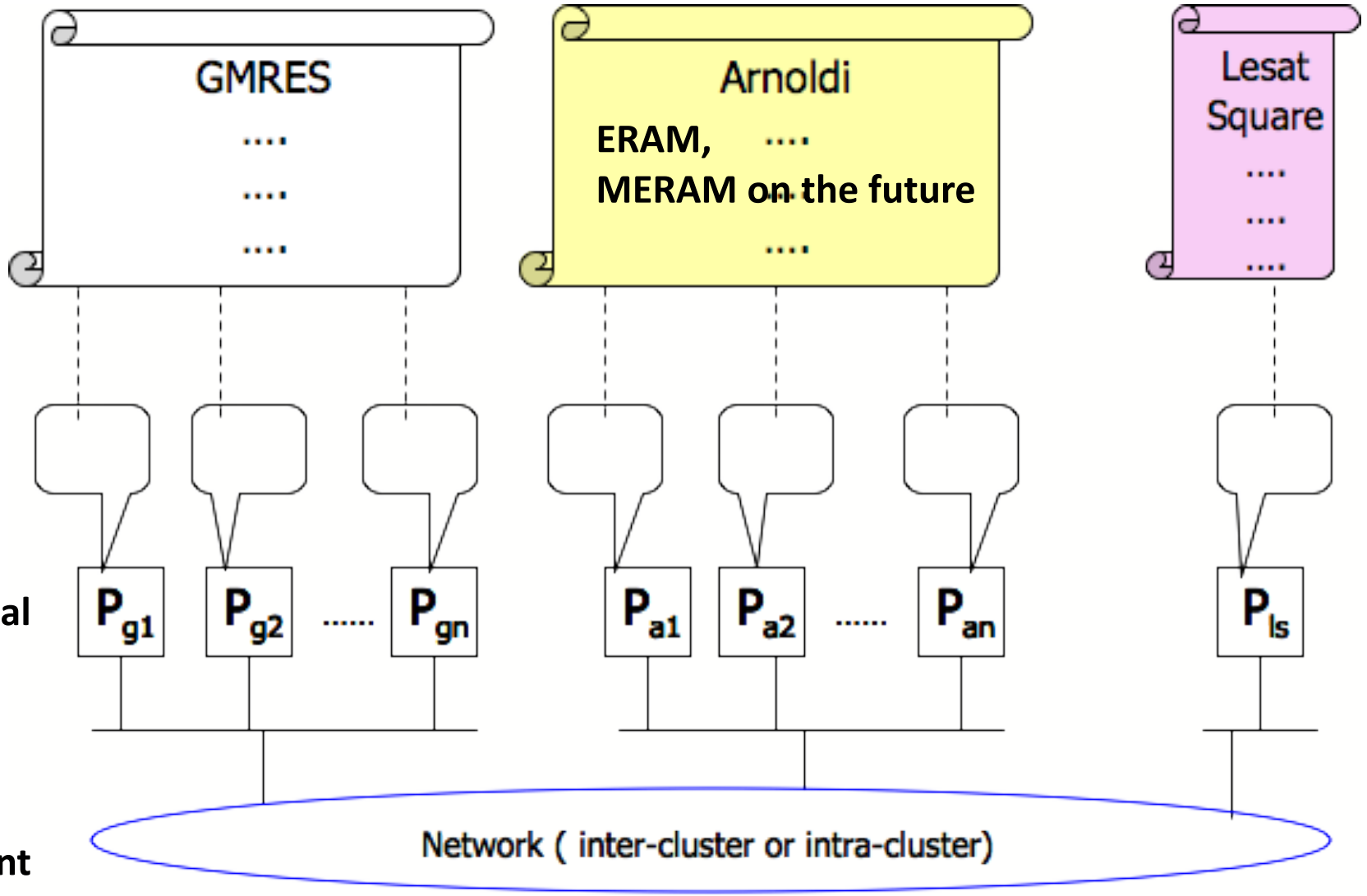
Figure: Eigenvalues, convex hull and ellipse





$$r_d = \sum_{i=1}^{k} R_d(\lambda_i)\rho_i u_i + \sum_{i=k+1}^{n} R_d(\lambda_i)\rho_i u_i$$

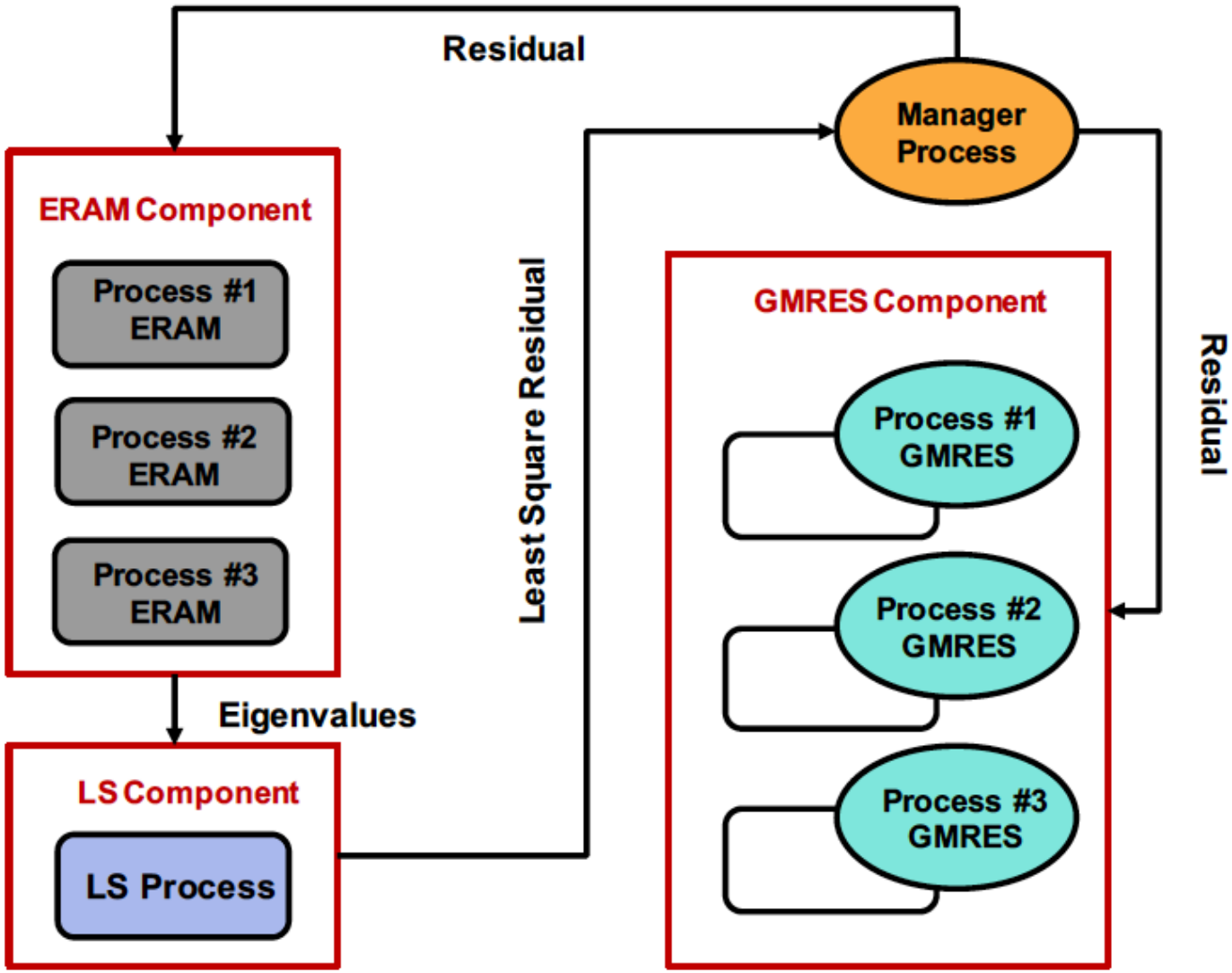
Asynchronous Restarted Krylov Methods
"Unite and Conquer GMRES/LS-ERAM" (UCGLE)



No any global operations

and

Fault Tolerant



…/…

Haiwu He, Guy Bergère, and Serge Petiton, Computational Math. Appl., 2006

…/…

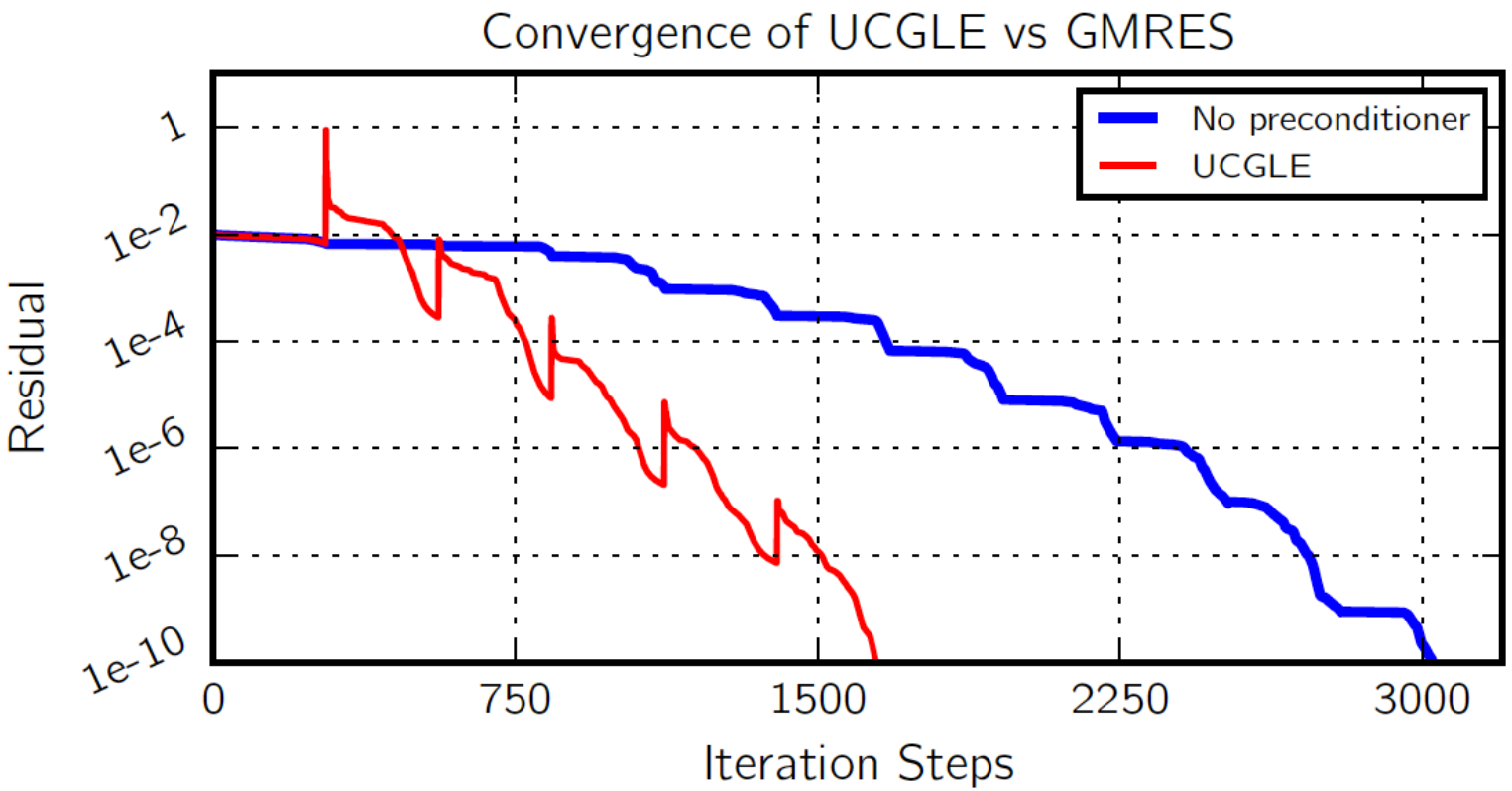
Ye Zhang, Guy Bergère, and Serge Petiton, LNCS, Springer Verlag, 2008

…/…

Least Squares Method

Least Squares method residual

$$r = (R_k(A))^{\iota} r_0 = \sum_{i=1}^{m} \rho((R_k)(\lambda_i)^{\iota}) u_i + \sum_{i=m+1}^{n} \rho((R_k)(\lambda_i)^{\iota}) u_i$$



Outline

- Introduction
- The GMRES-LS/ERAM method
- **The Distributed and Parallel "Unite and Conquer GMRES/LS ERAM" method (UCGLE)**
- Numerical Experiments
- Conclusion

Solving Sequence of Linear Systems

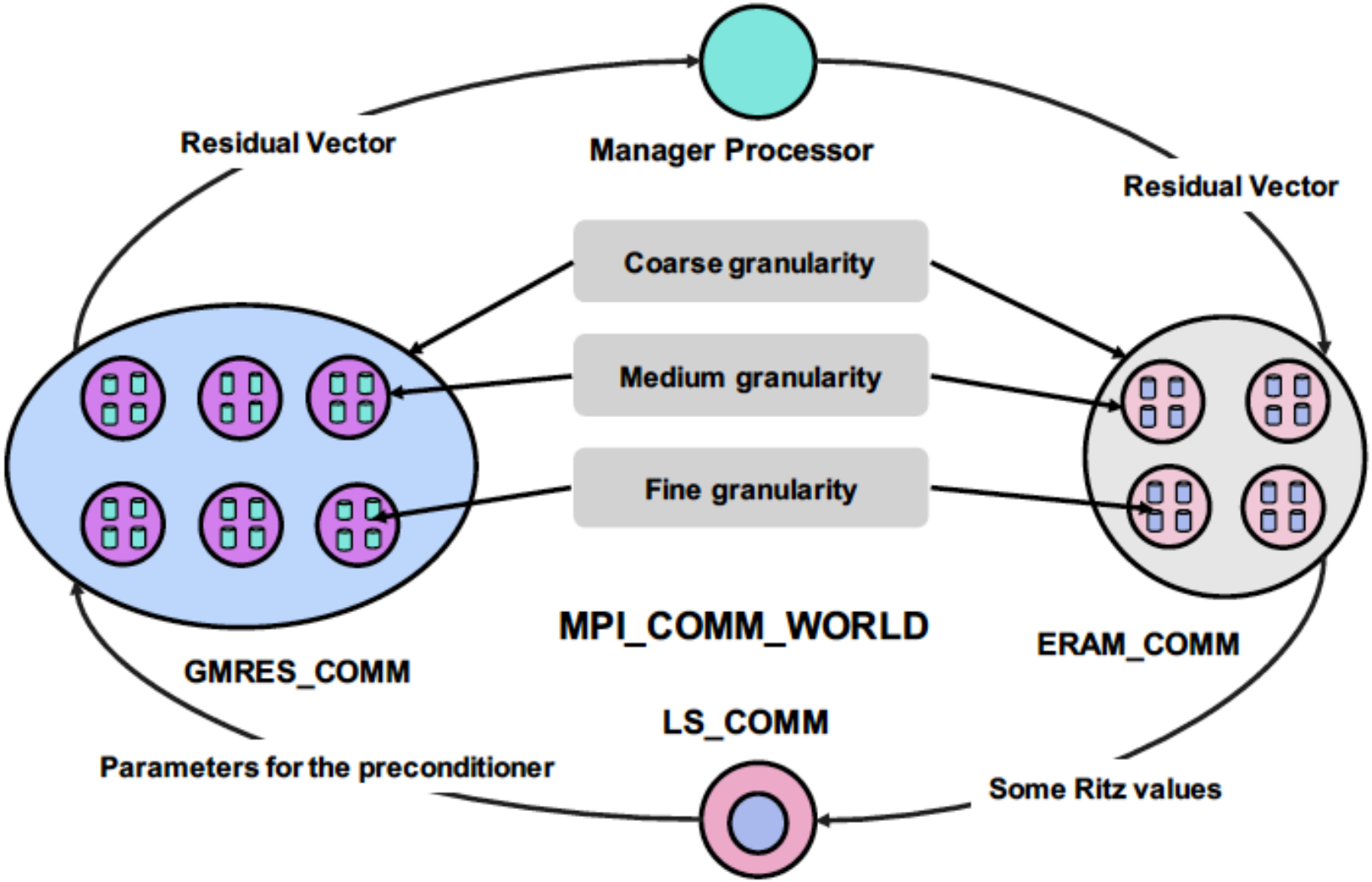
- We still compute the eigenvalues between linear system resolutions. Then, we have more eigenvalues with better accuracies, and them the convergence are fasters

- The new initial vectors are computed with LS method using the eigenvalues previously computed. Then, we have a faster convergence since the first GMRES.

- All communications are asynchronous and overlapped by computation

- We may use software from libraries and we have “just” to develop an engine to manage the different communication and launch the different methods.

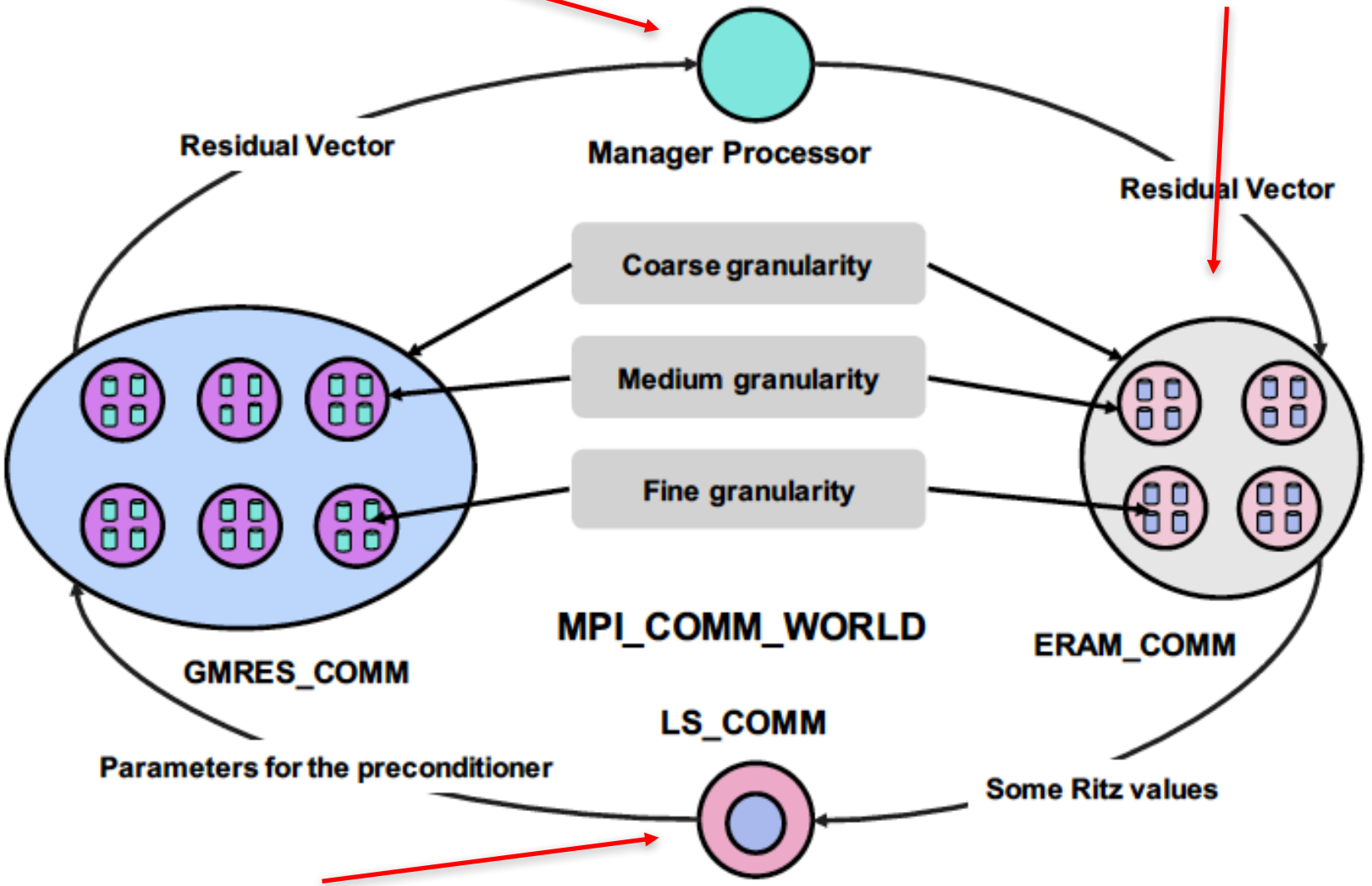
After we solve a first linear system



After we solve a first linear system, to start to solve the following linear systems (and the ERAM method never stop) :

The engine manage PETSc software and communications

We still compute eigenvalues



We compute a new initial guess using the LS method

Outline

- Introduction
- The GMRES-LS/ERAM method
- The Distributed and Parallel “Unite and Conquer GMRES/LS ERAM” method (UCGLE)
- **Numerical Experiments**
- Conclusion

Recent results : we compare 5 methods on the Tianhe 2(A) computer using PETSc

- GMRES: classic restarted GMRES;
- GMRES+SOR: GMRES with SOR preconditioner
- GMRES+Jacobi: GMRES with Jacobi preconditioner
- UCGLE “without” computed initial guess: UCGLE without using previous obtained eigenvalues to generate an initial guess vector for the next systems
- UCGLE “with” computed initial guess: UCGLE using previous obtained eigenvalues to generate an initial guess vector for the next systems (LS method)

A given method “with total CPUs” means that the number of CPU is equals to the total CPU in UCGLE (so, CPUs for both GMRES and ERAM are included)

Tianhe 2 : Intel Xeon + KNC ; (2A : + Matrix accelerator). We run with

- **LS run on 1 processor, GMRES on 768 processors and ERAM on 384**

Our test didn’t use the KNC as PETSc is really not efficient using these accelerators

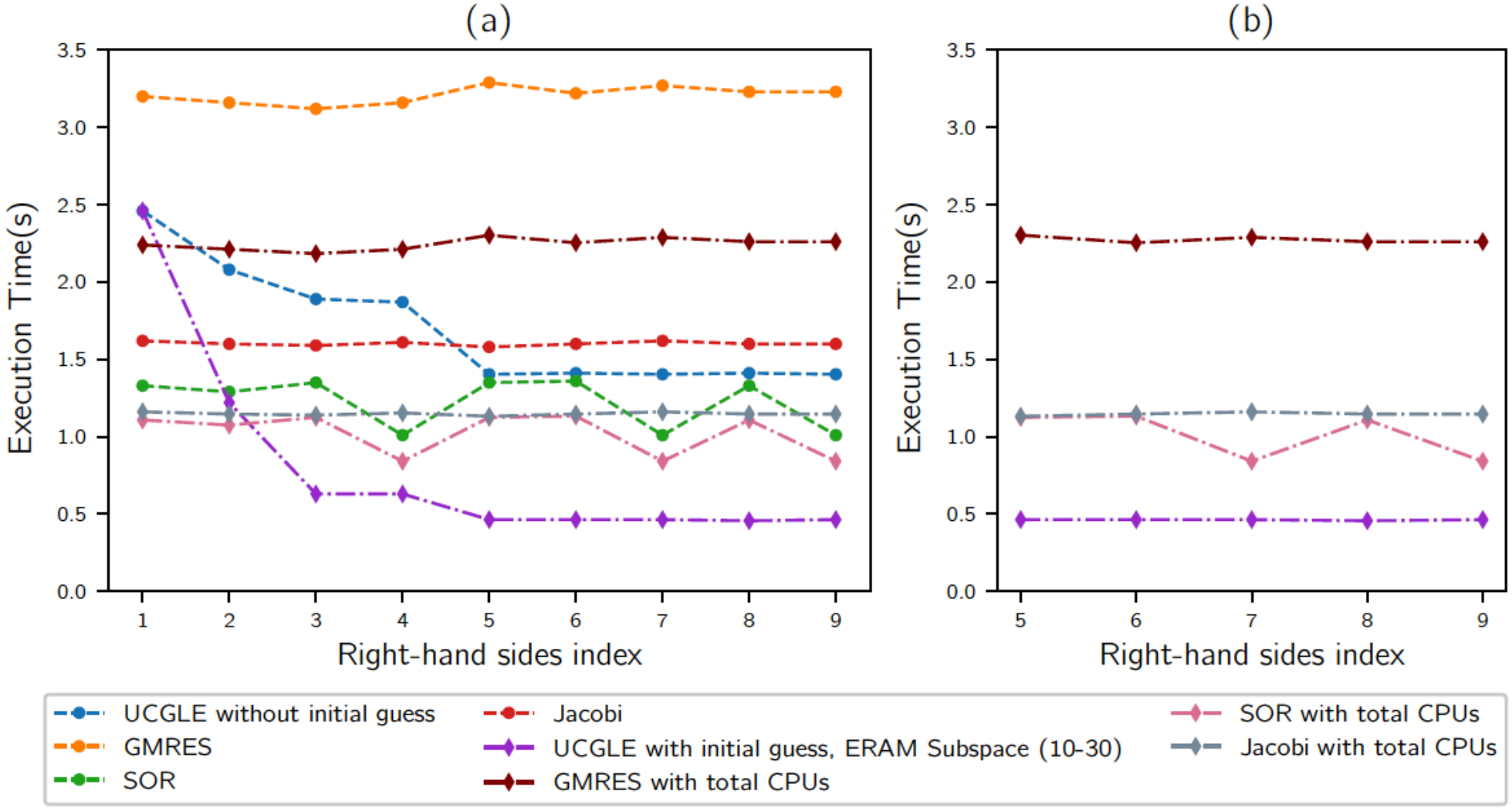
3 tests using large sparse matrices, generated by our "Scalable Matrix Generator with Given Spectra" (SMG2S)

(https://smg2s.github.io or https://github.com/SMG2S)

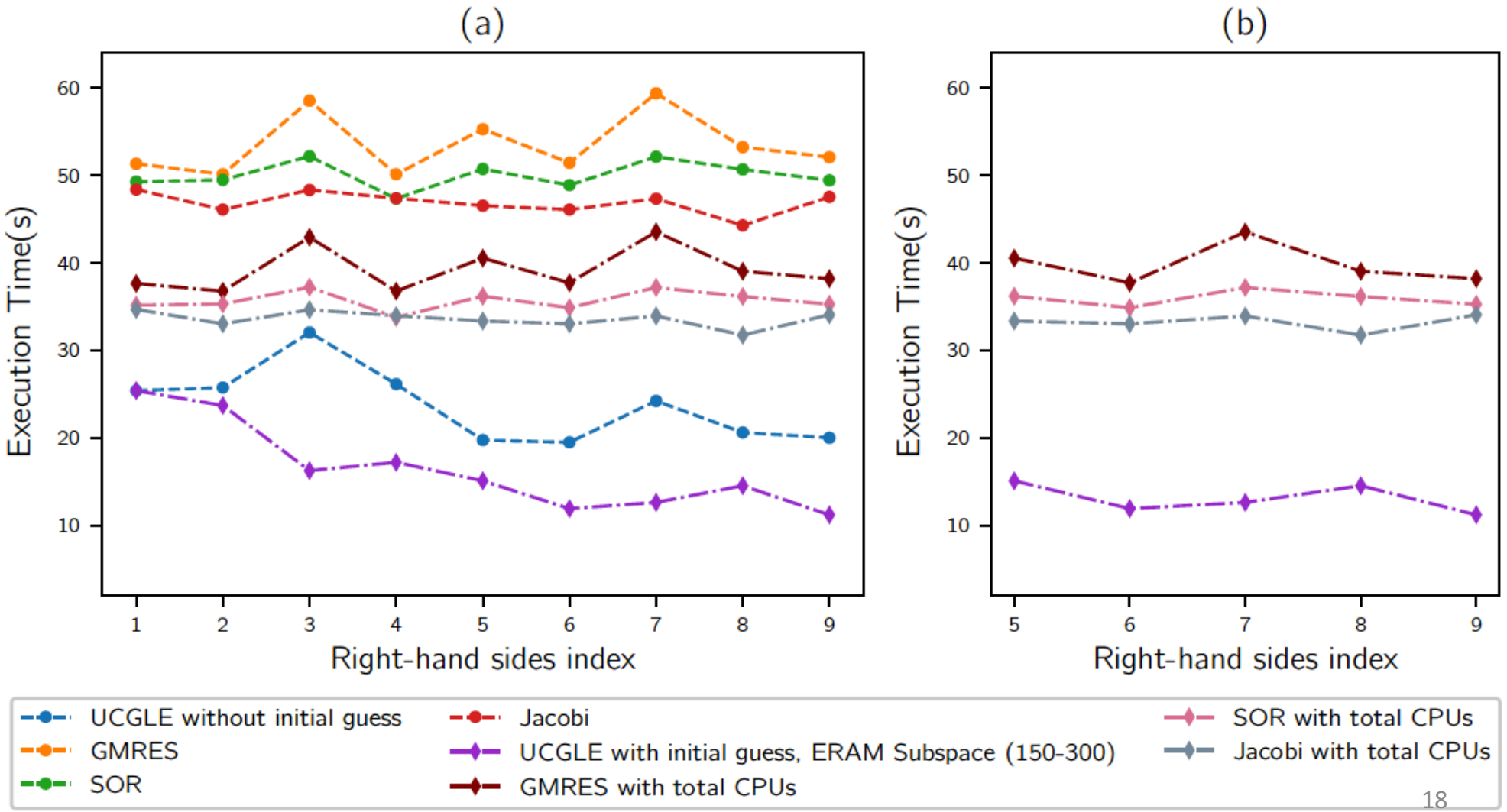
Matrix Mat1, Mat 2 and Mat 3 : 15,72 Millions, random spectra, nnz/row = 10

- Test 1: GMRES subspace = 30, ERAM subspace = 10, 20, 30 for the first three tests, and keeps 30 for the remaining systems in this test, and l' = 30 (parameter l' means the number of times that polynomial applied on the residual to generate new initial guess vector for current system using previous eigenvalues)
- Test 2: GMRES subspace = 300; ERAM subspace = 100, 150, 200, and keeps 200 for the remaining systems in this test, l' = 30
- Test 3: GMRES subspace = 150, ERAM subspace keeps to be 200, l'=20,30,40 for the 2nd, 3^{rd} and 4^{th} systems in this tests, for the remaining systems, l' keeps to be 40

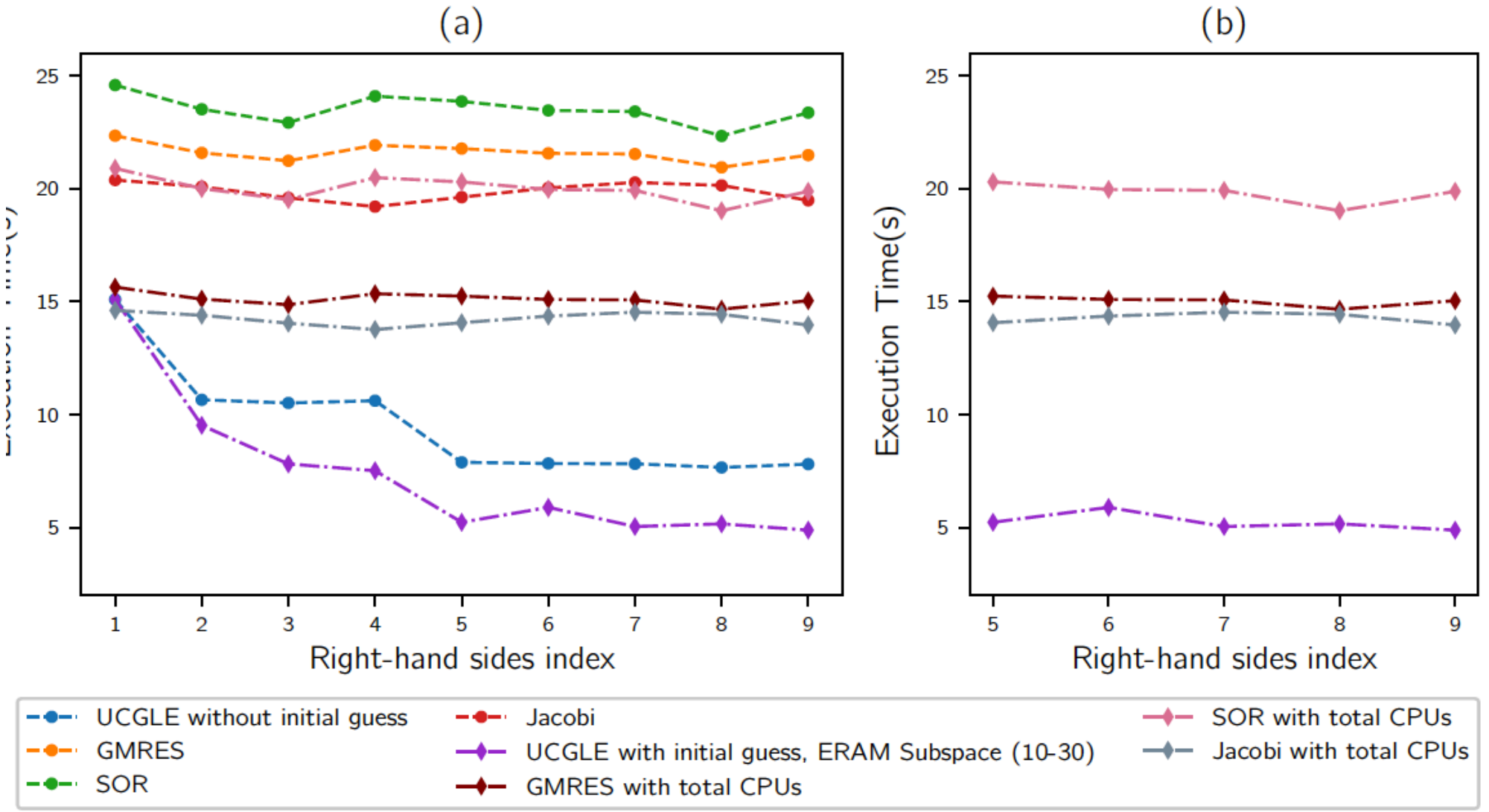
Mat 1
GMRES subspace size = 30
ERAM subspace size : 10, 20 and 30
l = 10 for each LS, l' = 30 for LS to compute new initial vector



Mat 2
GMRES subspace size = 300
ERAM subspace size = 100, 150 and 200
l = 10 iterations, each LS, l' = 30, for LS to compute new initial vector



Mat 3
GMRES subspace size = 150
ERAM subspace size = 200
l = 10, l’ = 20 (2), 30 (3) , 40 (4 and after)



Outline

- Introduction
- The GMRES-LS/ERAM method
- The Distributed and Parallel “Unite and Conquer GMRES/LS ERAM” method (UCGLE)
- Numerical Experiments
- **Conclusion**

Conclusion

- **Efficient method to solve sequence of linear system, fault tolerant and adapted for future extreme scale computers : with two levels or parallelism**
- We are testing UCGLE with several ERAM, or IRAM, (MERAM or MIRAM) to compute more eigenvalues, we are planning to use the Sakurai-Sugiura method to compute different parts of the spectrum
- We are testing multi right hand vector linear system problems (mixing Block-GMRES and UCGLE)
- We may use computed eigenvalues and use Deflated GMRES
- We are testing UCGLE with **Trilinos** and using GPU and other accelerators
- Using our matrix generator we may do a lot of experiments to analyze the convergence with respect to the a given spectrum.