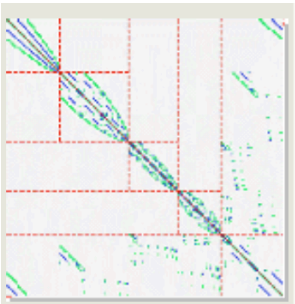


International Conference On Preconditioning Techniques For Scientific And Industrial Applications

July 31 - August 2, 2017
Vancouver, Canada

#2	15:30 - 17:30	HPC	Petiton, Emad, Chevalier, Kaman	Serge Petiton

High Performance Computing

A Distributed and Parallel Asynchronous Unite and Conquer Method to Solve Large Scale Non-Hermitian Linear Systems

Serge G. Petiton [1]

[1]University Lille 1, France (Serge.Petiton@univ-lille1.fr)

Spectral Graph Analysis with Unite and Conquer Approach

Nahid Emad [1]

[1]University of Versailles, France (Nahid.Emad@uvsq.fr)

Algorithms for Multi-Criteria Mesh Partitioning

Cédric Chevalier [1]

[1]CEA, France (cedric.chevalier@cea.fr)

Strategies for the Development of Efficient, High-Order Accurate Residual Distribution Schemes for the Solution of Compressible Flow Problems

Tulin Kaman [1]

[1]University of Zurich, Switzerland (tulin.kaman@math.uzh.ch)

A Distributed and Parallel Asynchronous Unite and Conquer Method to Solve Large Scale Non-Hermitian Linear Systems

Serge G. Petiton[1,2], Xinzhe Wu[1,2] and Tao Chang[1]

[1] CNRS : Maison de la Simulation, Paris-Saclay, and CRISTAL, Lille, France
[2] University of Lille, Sciences and Technologies, France

Outline

- Introduction, toward Exascale Computation and beyond
- Distributed and Parallel Programming for Extreme Computing
- The Asynchronous Unite and Conquer GMRES/LS-ERAM (UCGLE) method
- Experimentations, evaluation and analysis
- Conclusion and perspectives

Outline

- **Introduction, toward Exascale Computation and beyond**
- Distributed and Parallel Programing for Extreme Computing
- The Asynchronous Unite and Conquer GMRES/LS-ERAM (UCGLE) method
- Experimentations, evaluation and analysis
- Conclusion and perspectives

Toward extreme computing, some correlated goals

- Minimize the global computing time,
- Accelerate the convergence,
- Minimize the number of communications (optimized Ax, asynchronous comp, communication compiler and mapper,….)
- Minimize the number of longer size scalar products, and reductions
- Minimize memory space, cache optimization….
- Select the best sparse matrix compressed format,
- Mixed arithmetic
- Minimize energy consumption
- Fault tolerance, resilience

Toward extreme computing, some correlated goals

- Minimize the global computing time,
- Accelerate the convergence,
- *Minimize the number of communications (optimized Ax, asynchronous comp, communication compiler and mapper,….)*
- Minimize the number of longer size scalar products, and reductions
- *Minimize memory space, cache optimization….*
- *Select the best sparse matrix compressed format*,
- Mixed arithmetic
- *Minimize energy consumption*
- Fault tolerance, resilience

New methods, and especially new or adapted preconditioning, may propose some solutions for some of these challenges. Nevertheless, the programming paradigms would change an we have to consider future exascale, and beyond, supercomputer as distributed platforms with parallel clusters.

Toward extreme computing, some correlated goals

- Minimize the global computing time,
- Accelerate the convergence,
- *Minimize the number of communications (optimized Ax, asynchronous comp, communication compiler and mapper,….)*
- Minimize the number of longer size scalar products, and reductions
- *Minimize memory space, cache optimization….*
- *Select the best sparse matrix compressed format*,
- Mixed arithmetic
- *Minimize energy consumption*
- Fault tolerance, resilience

Preconditioning

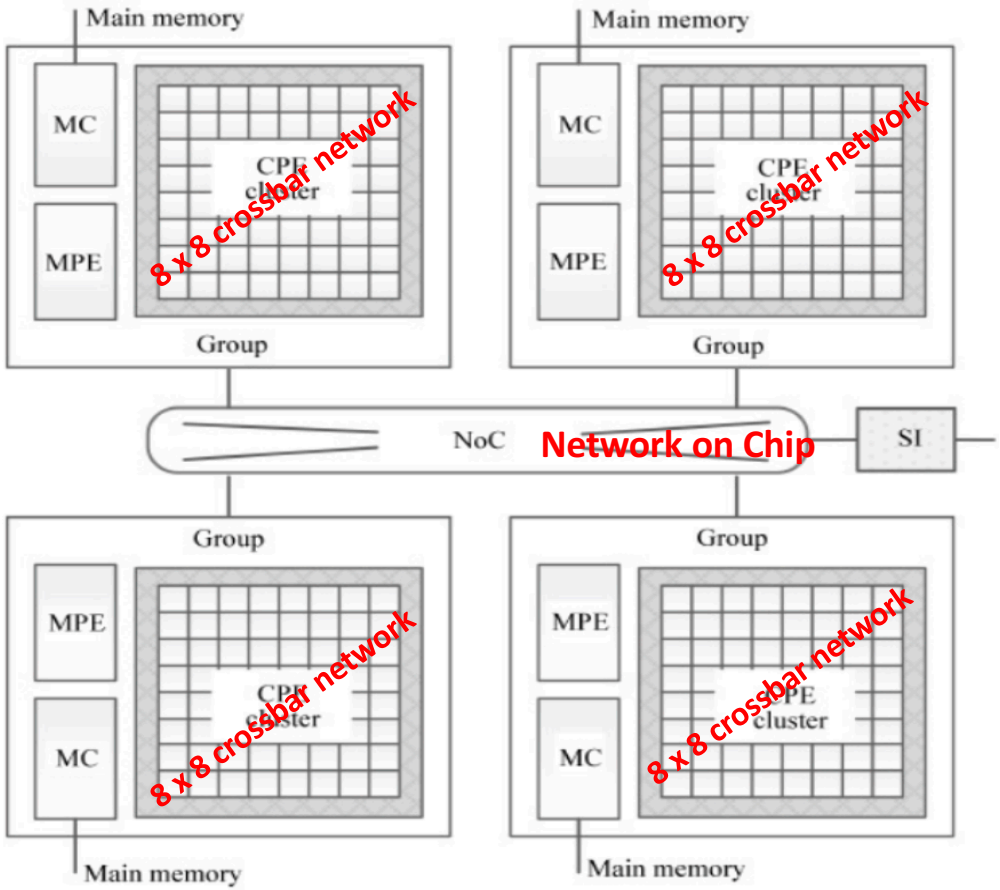
Unite and conquer : Reductions and spreads on subsets of the machine

Unite and conquer

New methods, and especially new or adapted preconditioning, may propose some solutions for some of these challenges. Nevertheless, the programming paradigms would change an we have to consider future exascale, and beyond, **supercomputer as distributed platforms with parallel clusters**.

Even on a processor : example of the sunway TaihuLight MPP

Each group of cores is composed by a management unit (MPE) and 64 computing units (CPE) structured on a 8 x 8 grid, connected by a network on chip (NOC) : ***Distributed computing inside each chip***



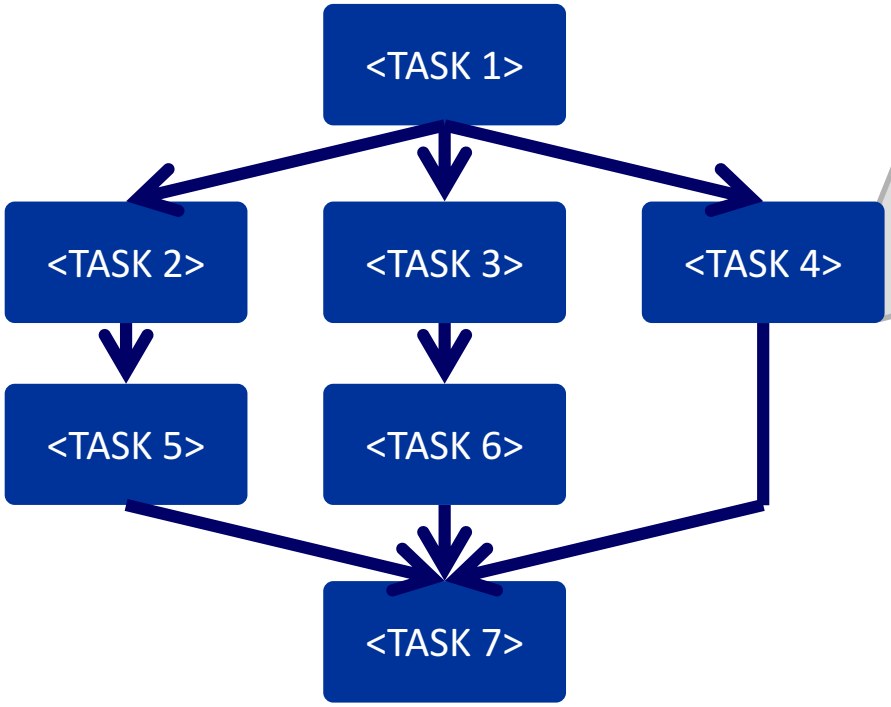
On each SW 26019 processor, made in China, we have 4 CPEs (64 x 4 = 256 cores) and 4 MPE - e.g. ***260 cores*** – and 4 memory controllers (MC). Each MC has 8 Gigabytes of memory.

Outline

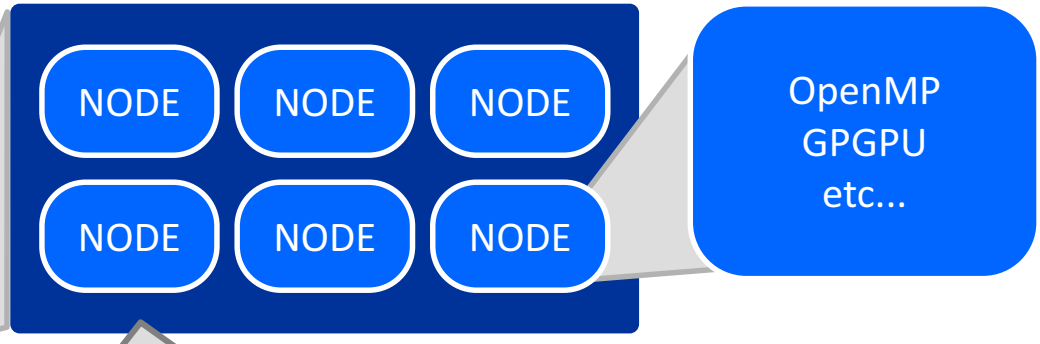
- Introduction, toward Exascale Computation and beyond
- **Distributed and Parallel Programing for Extreme Computing**
- The Asynchronous Unite and Conquer GMRES/LS-ERAM (UCGLE) method
- Experimentations, evaluation and analysis
- Conclusion and perspectives

Multi-Level Parallelism Integration: YML-XMP

N dimension graphs available



YML provides a workflow programming environment and high level graph description language called YvetteML

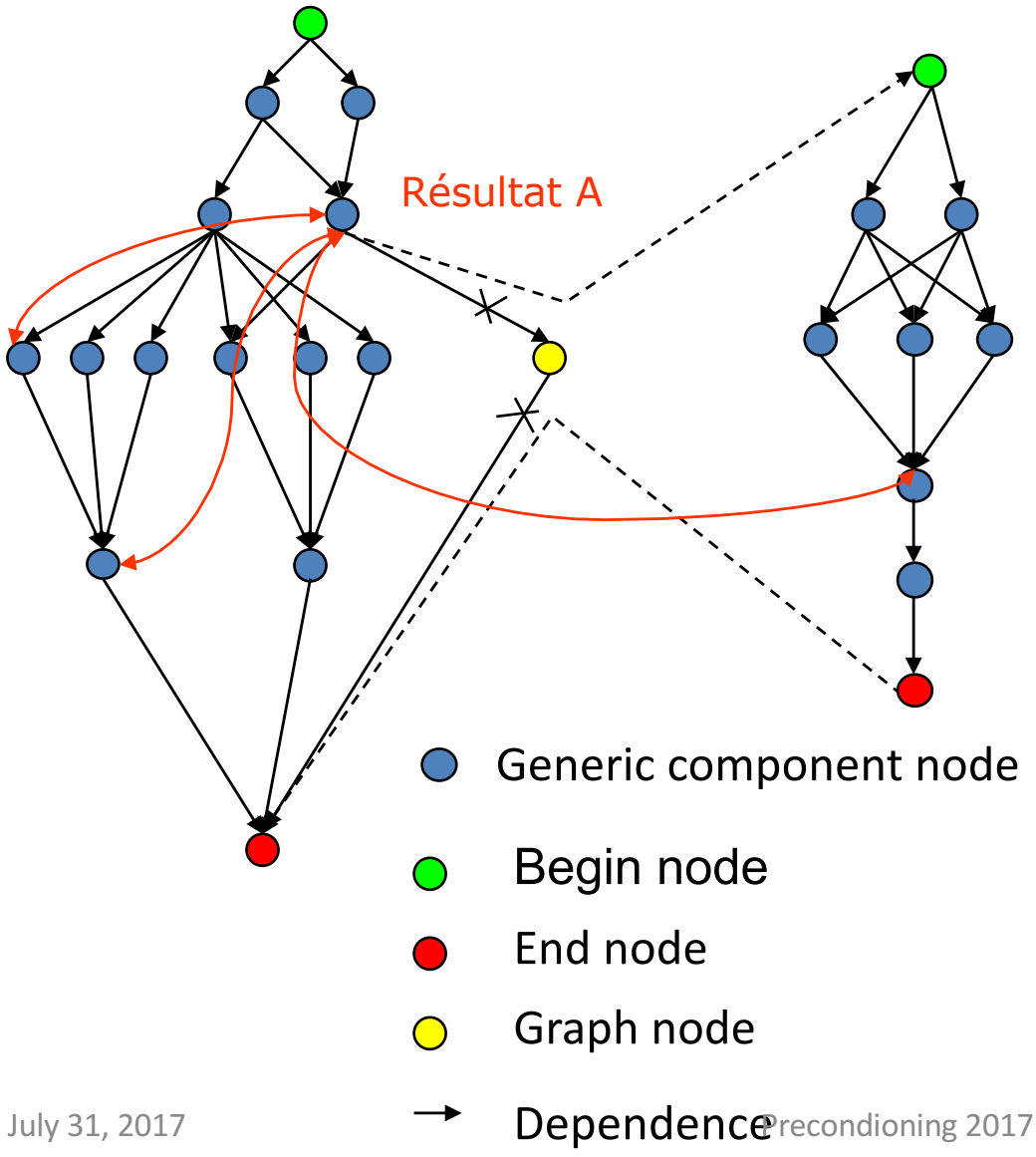



```
for(i=0;i<n;i++){
    for(j=0;j<n;j++){
       tmp[i][j]=0.0;
#pragma xmp loop (k) on t(k)
         for(k=0;k<n;k++){
            tmp[i][j]+=(m1[i][k]*m2[k][j]);
}}}
#pragma xmp reduction (+:tmp)
```

Each task is a parallel program over several nodes.
XMP language can be used to descript parallel program easily!

YML/XMP/StarPu experiments on T2K in Japan, project FP3C

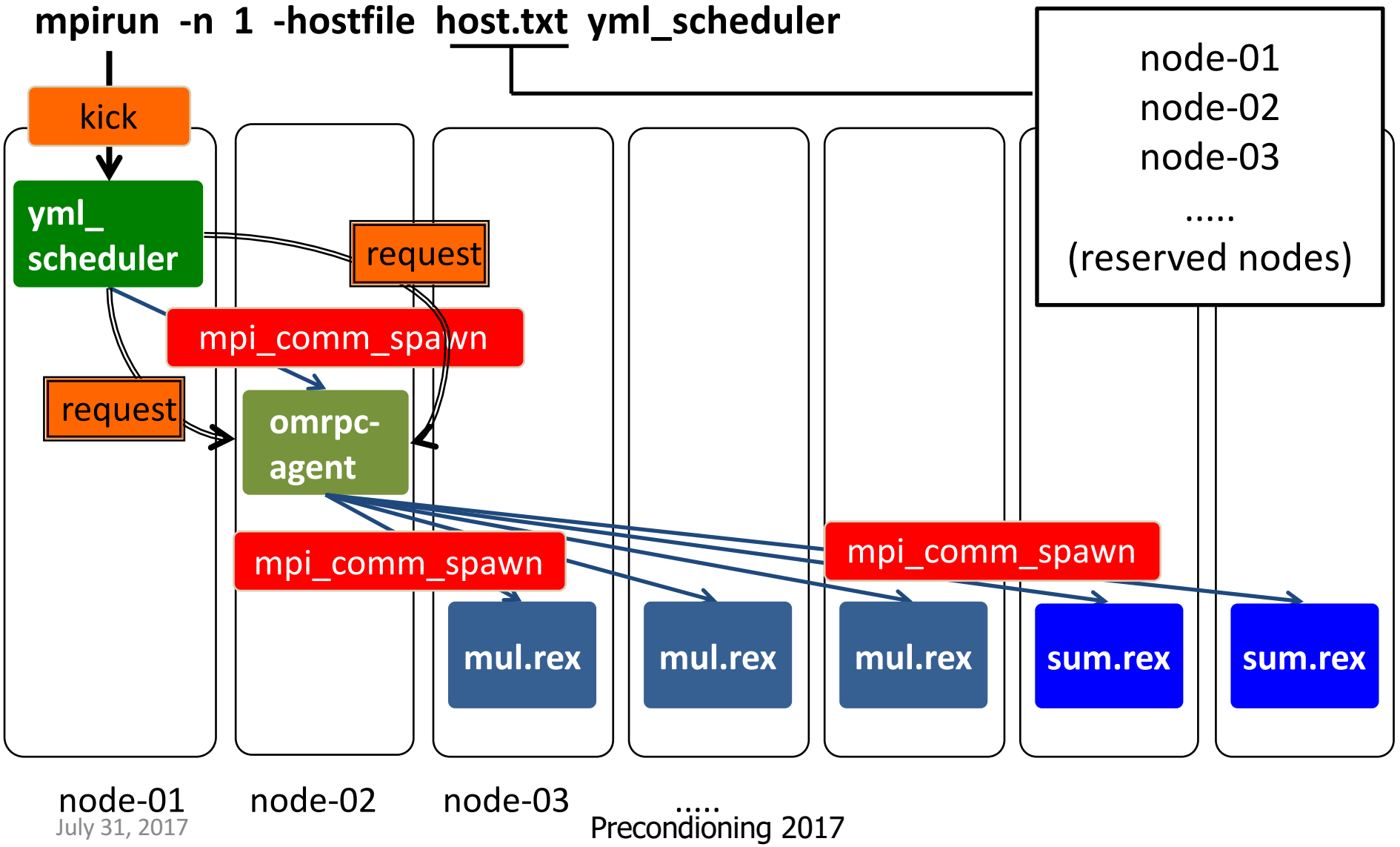
Components/Tasks/Containers Graph Dependency




```
par
  compute tache1(..);
  notify(e1);
//
  forall i=1,n
      compute tache2(.i,.); migrate
matrix(..);
   notify(e2(i));
  end forall
//
  wait(e1 and e2(1);
  Par
    compute tache3(..);
    signal(e3);
  //
    compute tache4(..);
    signal(e4);
  //
    compute tache5(..); control robot(..);
    signal(e5); visualize mesh(…) ;
  end par
//
  wait(e3 and e4 and e5);
  compute tache6(..);
  compute tache7(..);
end par
```

Slide : Miwako Tusji and Maxime Hugues

Experimentation on the K Computer, AICS, Japan



Outline

- Introduction, toward Exascale Computation and beyond
- Distributed and Parallel Programming for Extreme Computing
- **The Asynchronous Unite and Conquer GMRES/LS-ERAM (UCGLE) method**
- Experimentations, evaluation and analysis
- Conclusion and perspectives

The method, proposed by Youcef Saad in 1987 :

> Youcef Saad. Least squares polynomials in the complex plane and their use for solving nonsymmetric linear systems. *SIAM Journal on Numerical Analysis*, 24(1):155–169, 1987.

Consider a linear system of equations

$$Ax = b \tag{2}$$

Note that polynomial preconditioning of system (1), we can get the polynomial iteration

$$x_n = x_0 + P_n(A)r_0 \tag{3}$$

where x_0 is some intial approximation to the solution, r_0 the corresponding residu, and P_n a polynomial of degree $n-1$. We set a ploynomial of n degree R_n such that:

$$R_n(\lambda) = 1 - \lambda P_n(\lambda) \tag{4}$$

The residu of n^{th} steps of iteration r_n can be expressed as equation (4) with the constraint $R_n(0) = 1$

$$r_n = R_n(A)r_0 \tag{5}$$

Clearly, one wants to find a kind of polynomial which can minimize $||R_n(A)r_0||_2$,

Then, if we know the spectrum of A :

$$||r_n||_2 \leq ||r_0||_2 \max_{\lambda \in \sigma(A)} |R_n(\lambda)|$$

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

Or, if we just know a subset of the spectrum :

$$min\ max_{\lambda \in H} |R_n(\lambda)|$$

Suppose that the computed convex hull *convex* by Least Squares contains eigenvalues $\lambda_1, \cdots, \lambda_m$, the residual given by Least Square is

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

Clearly, one wants to find a kind of polynomial which can minimize $||R_n(A)r_0||_2$,

Then, if we know the spectrum of A :

$$||r_n||_2 \leq ||r_0||_2 \max_{\lambda \in \sigma(A)} |R_n(\lambda)|$$

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

Computed with LS

Or, if we just know a subset of the spectrum :

$$min\ max_{\lambda \in H} |R_n(\lambda)|$$

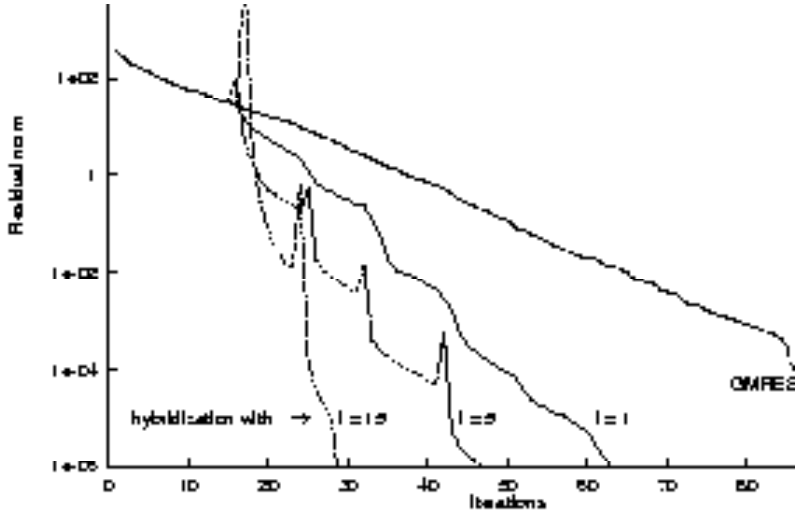
Computed with ERAM

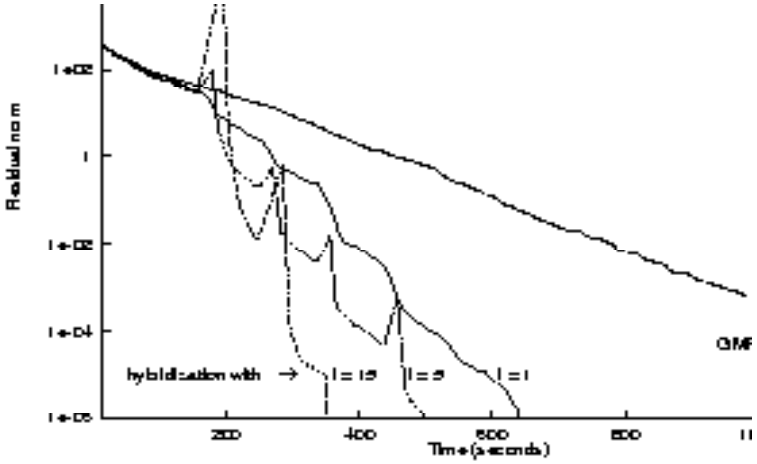
Suppose that the computed convex hull *convex* by Least Squares contains eigenvalues $\lambda_1, \cdots, \lambda_m$, the residual given by Least Square is

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

minimization

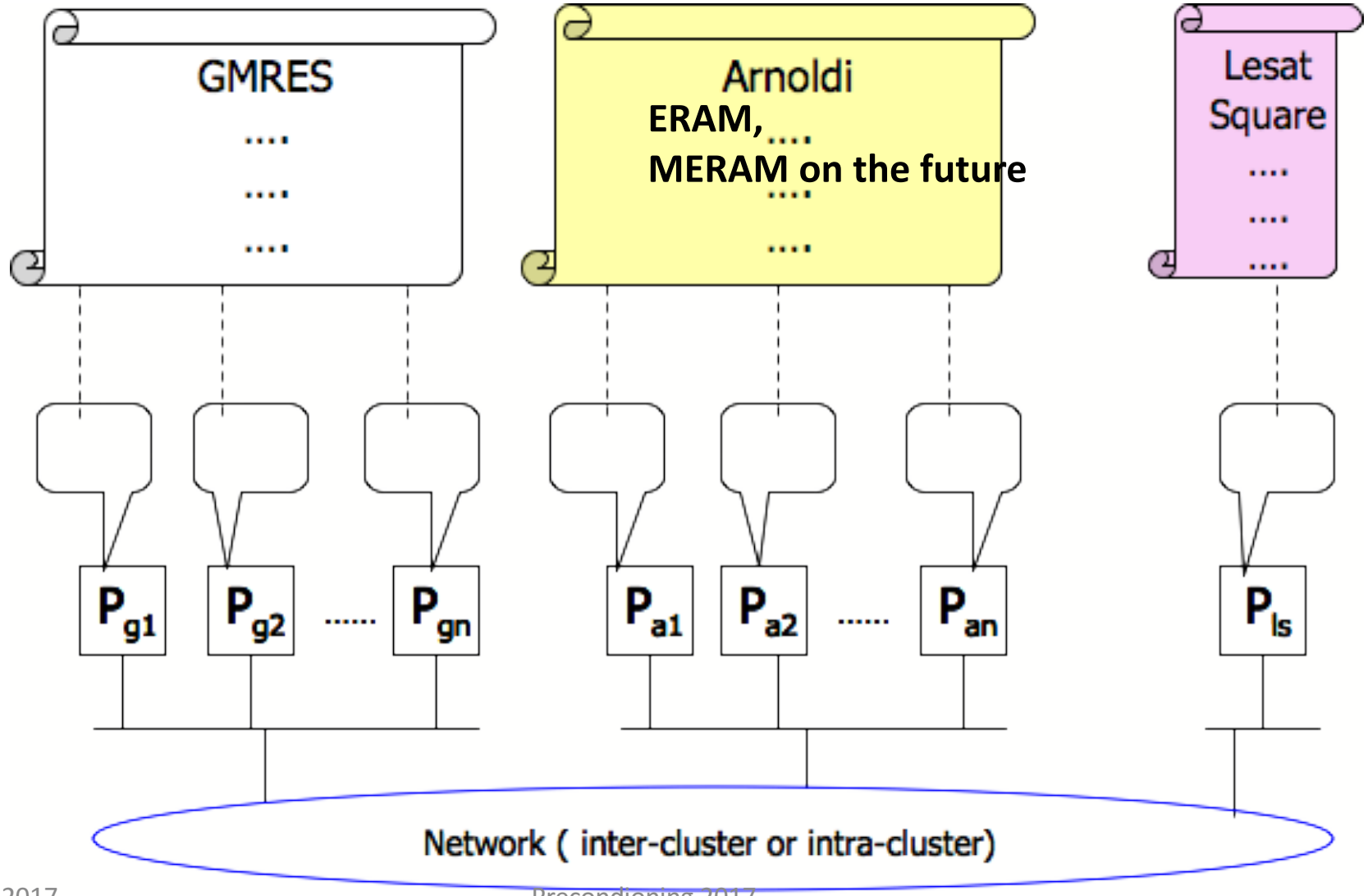
may increase


Residual norm
1e+02
1
1e-02
1e-04
1e-06
hybridization with →
l = 15
l = 5
l = 1
GMRES
0
10
20
30
40
50
60
70
80
Iterations

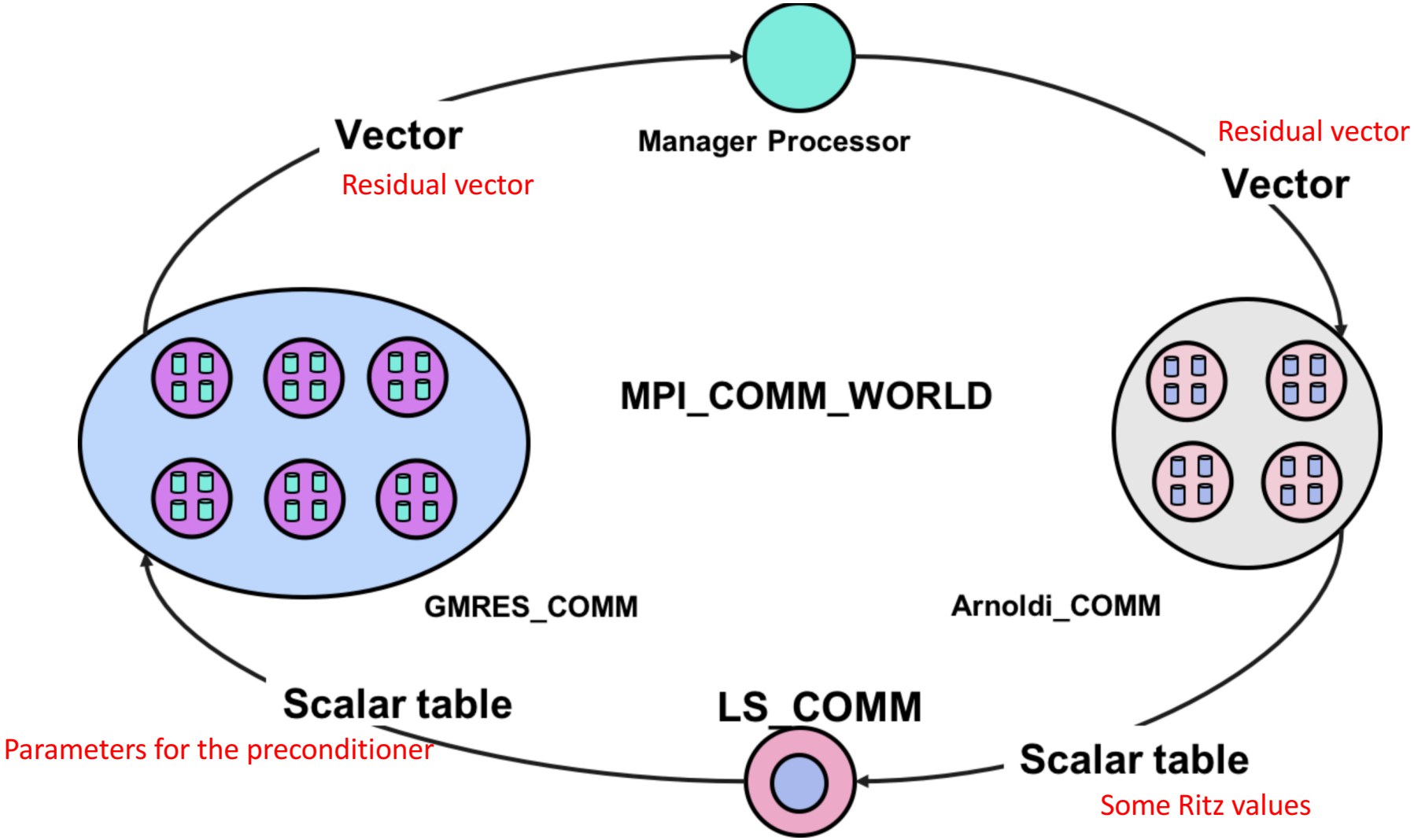

Residual norm
1e+02
1
1e-02
1e-04
1e-06
hybridization with →
l = 15
l = 5
l = 1
GMR
200
400
600
Time (seconds)

The Asynchronous Unite and Conquer GMRES/LS-ERAM Method (UCGLE)

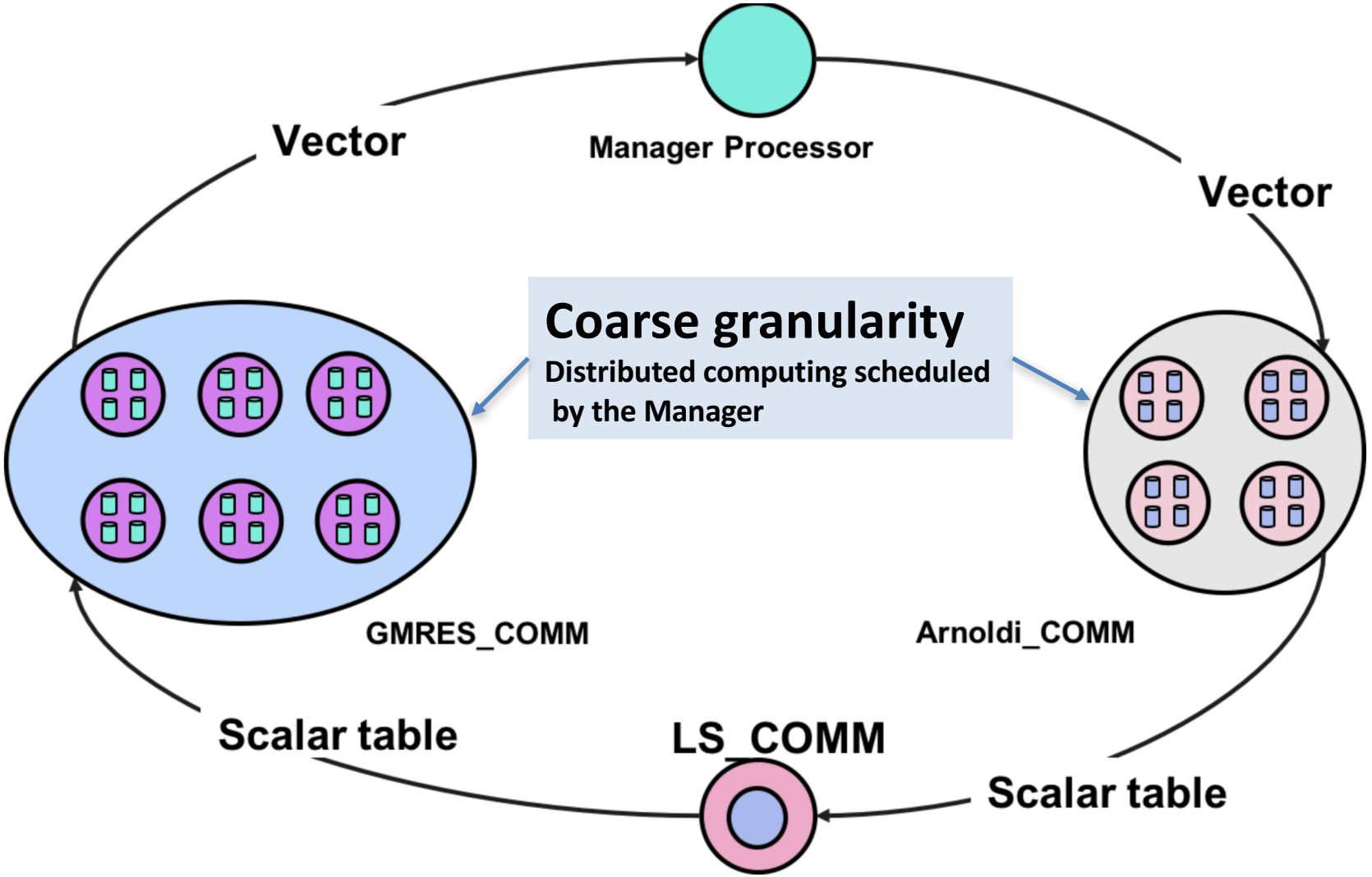
Collaboration with Azedine Essai, He Haiwu and Guy Bergère (U. Lille 1, CNRS) and Ye Zhang (Hohai Univ. Nanjing) , Salim Nahi (Maison de la simulation), and Pierre-Yves Aquilenti (TOTAL)



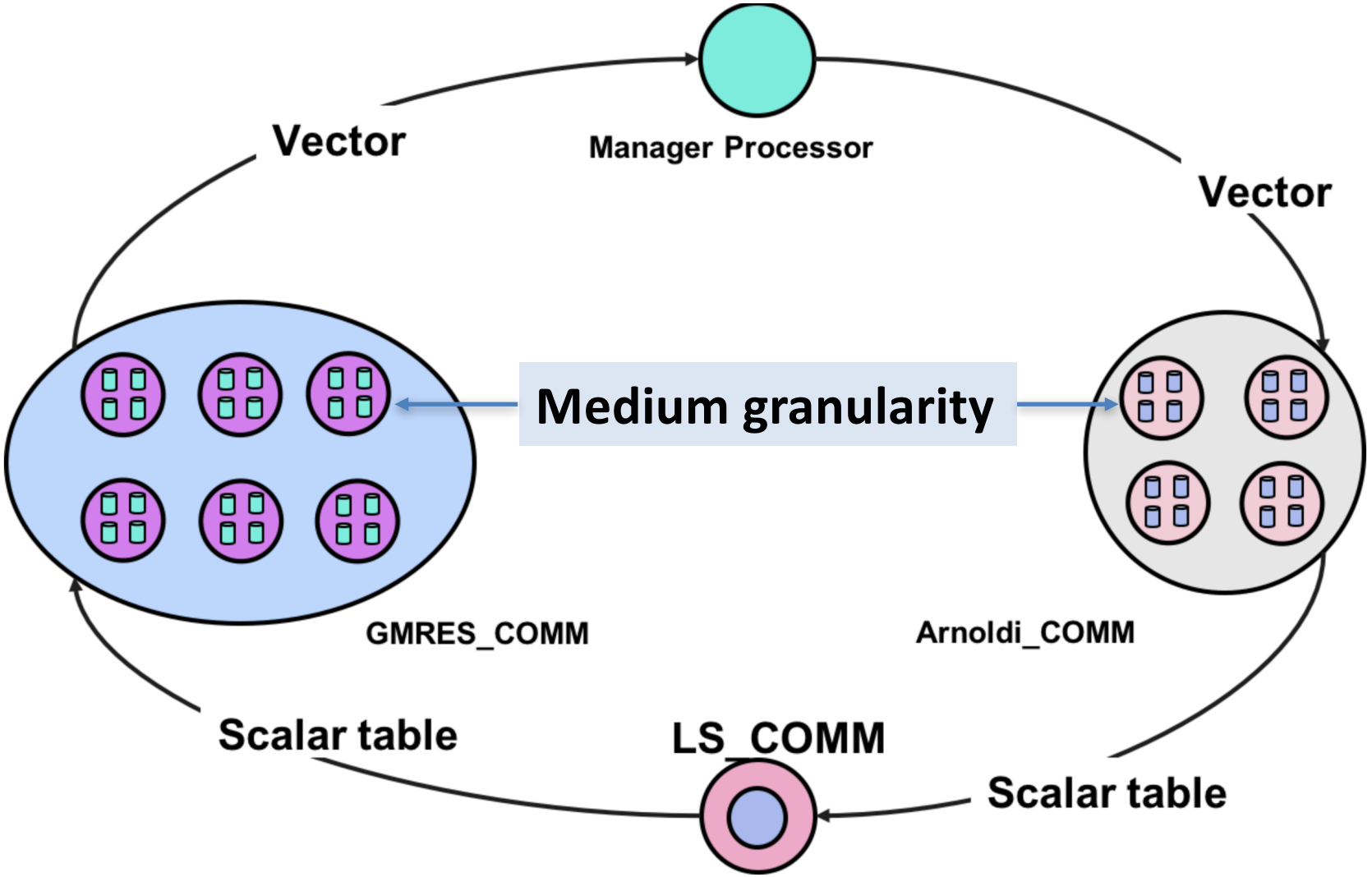
Software engine, orchestration of this unite and conquer method



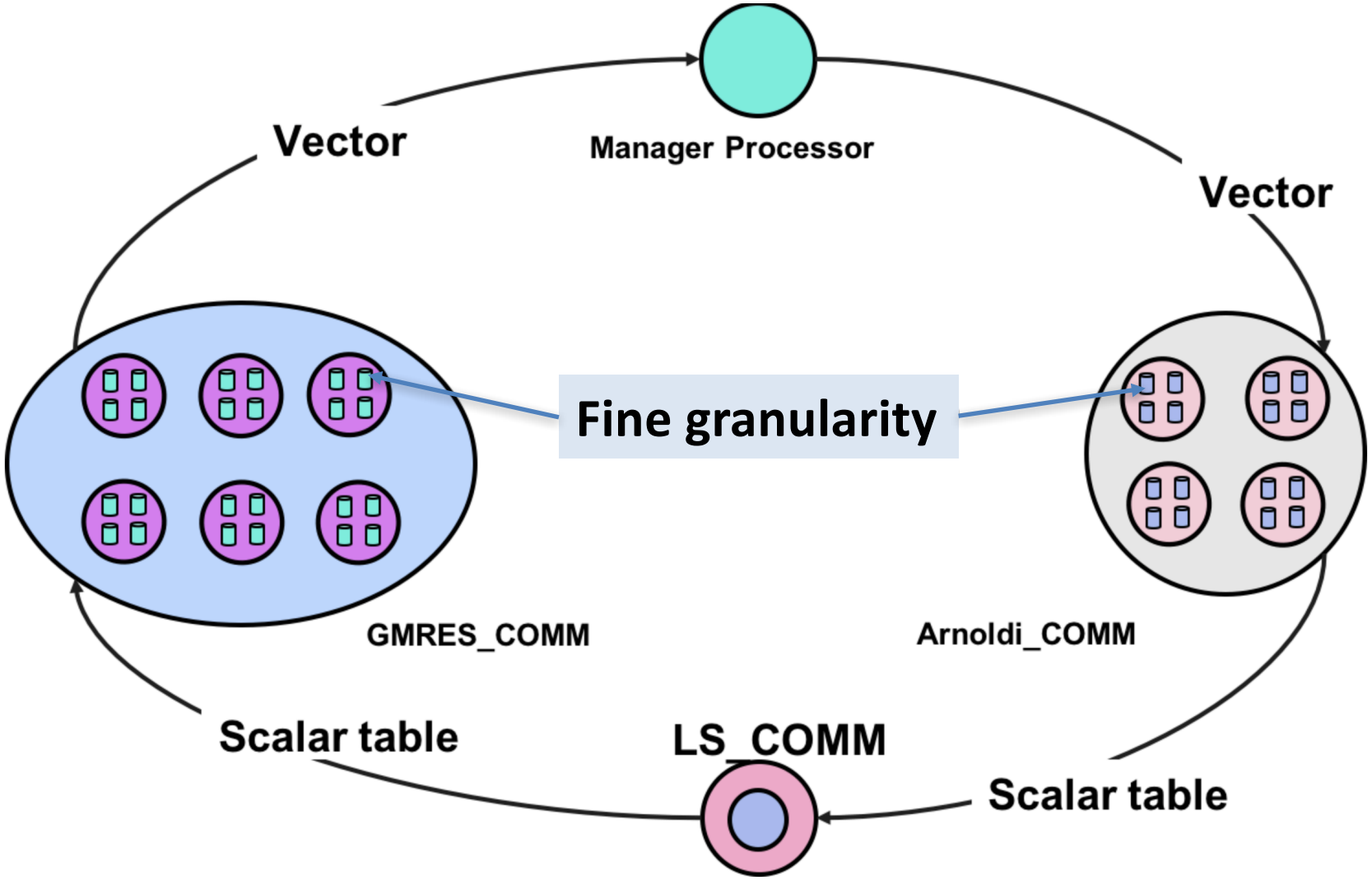
Software engine, orchestration of this unite and conquer method



Software engine, orchestration of this unite and conquer method

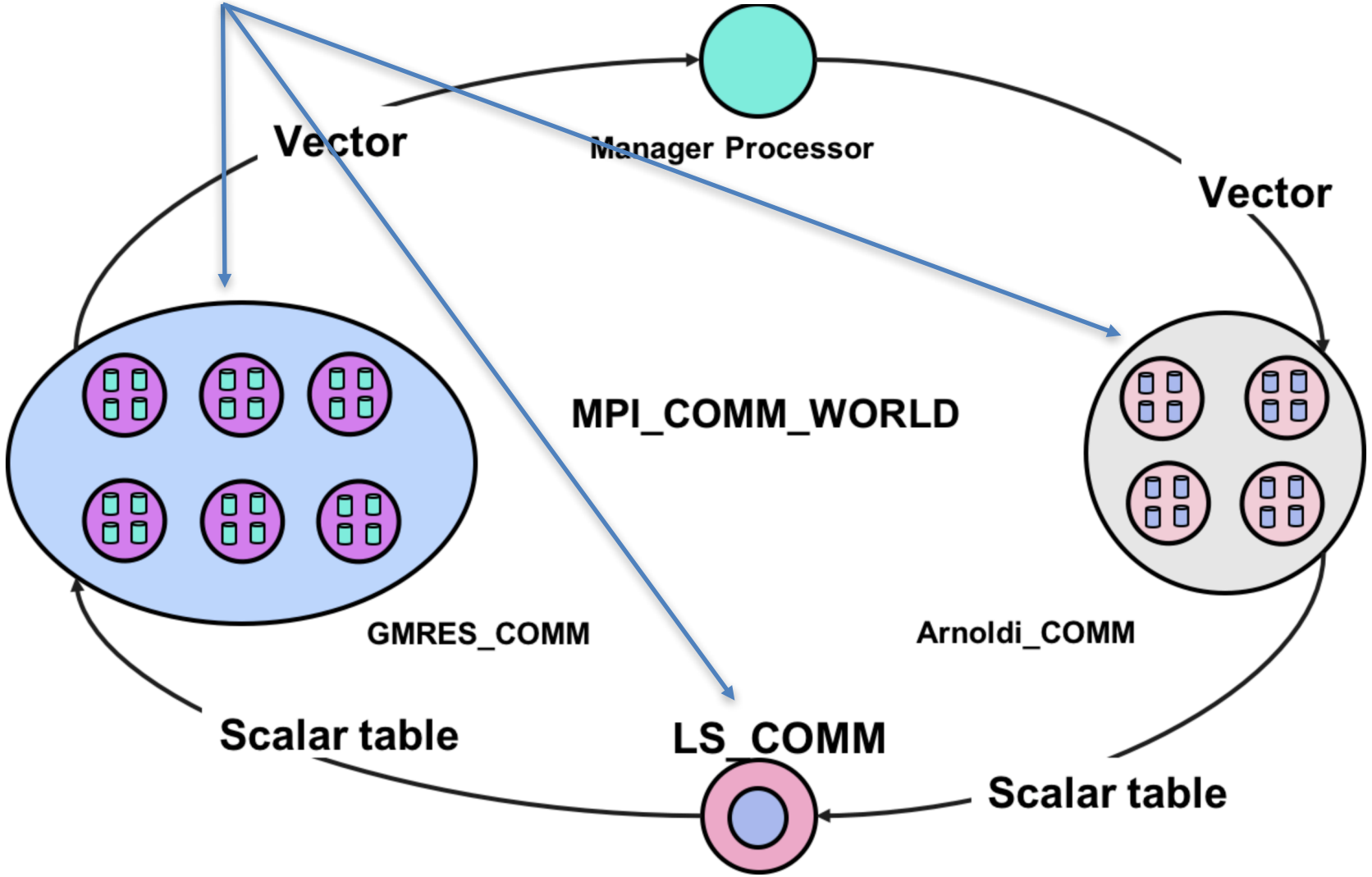


Software engine, orchestration of this unite and conquer method



With accelerator sometime associated to each nodes

Using PETSc and SLEPc only



Libraries are not often optimized for the most recent accelerators
Using PETSc and SLEPc assure the “portability”. **We developed the software engine, it is possible then to use any codes for GMRES, ERAM or LS**.

Several parameters :

I. GMRES Component
   * m_g: GMRES Krylov Subspace size
   * ϵ_g: absolute tolerance used for the GMRES convergence test
   * P_g: GMRES processors number
   * s_{use}: number of times that polynomial applied before taking account into the new eigenvalues
   * L: number of GMRES restarts before each time LS precondtioning

II. Arnoldi Component
   * m_a: Arnoldi Krylov subspace size
   * r: number of eigenvalues required
   * ϵ_a: convergence tolerance
   * P_a: Arnoldi processors number

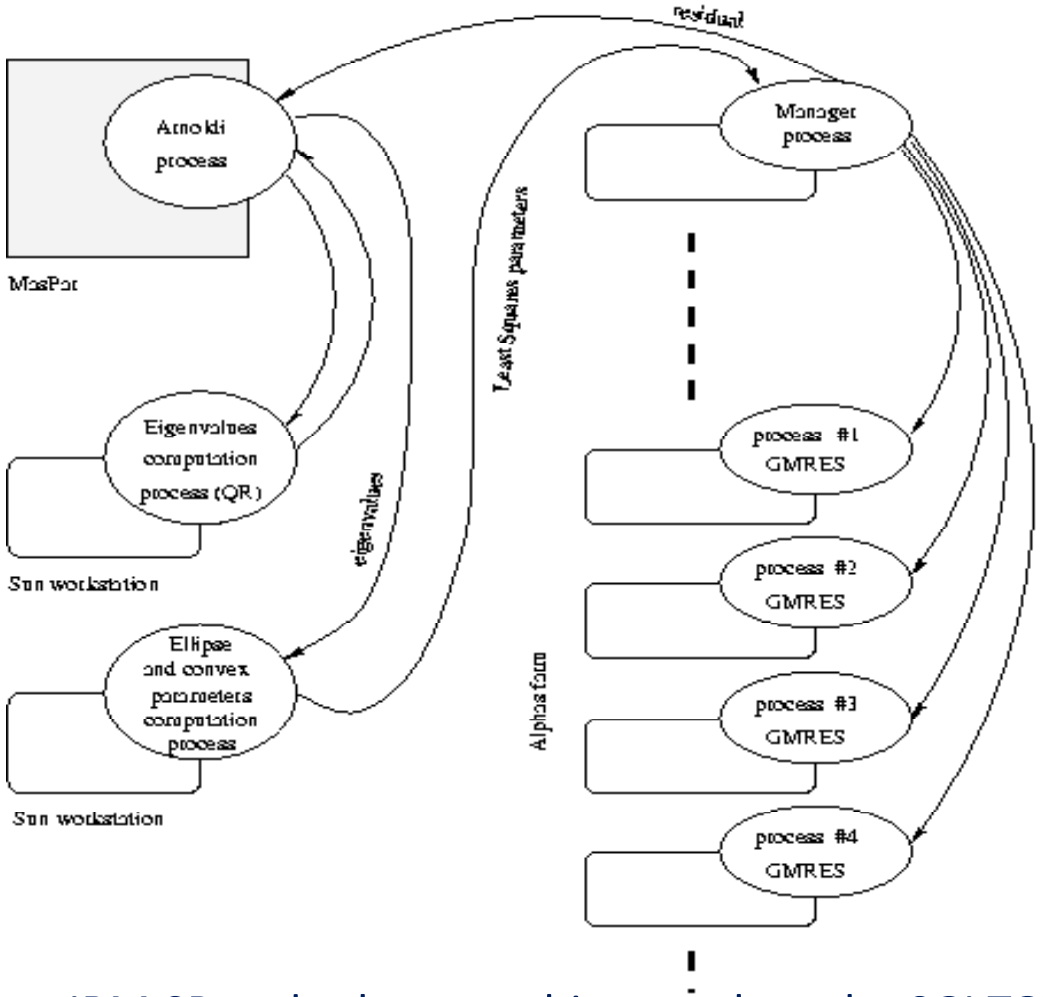
III. LS Component
   * d: Least Squares polynomial degree

We studied on the past auto-tuning at runtime of some parameters such as subspace sizes or number of vectors of the Krylov basis to orthogonalize, cf. publications with Pierre Yves Aquilenti (Total) and Takahiro Katagiri (U. Tokyo)

Outline

- Introduction, toward Exascale Computation and beyond
- Distributed and Parallel Progamming for Extreme Computing
- The Asynchronous Unite and Conquer GMRES/LS-ERAM (UCGLE) method
- **Experimentations, evaluation and analysis**
- Conclusion and perspectives

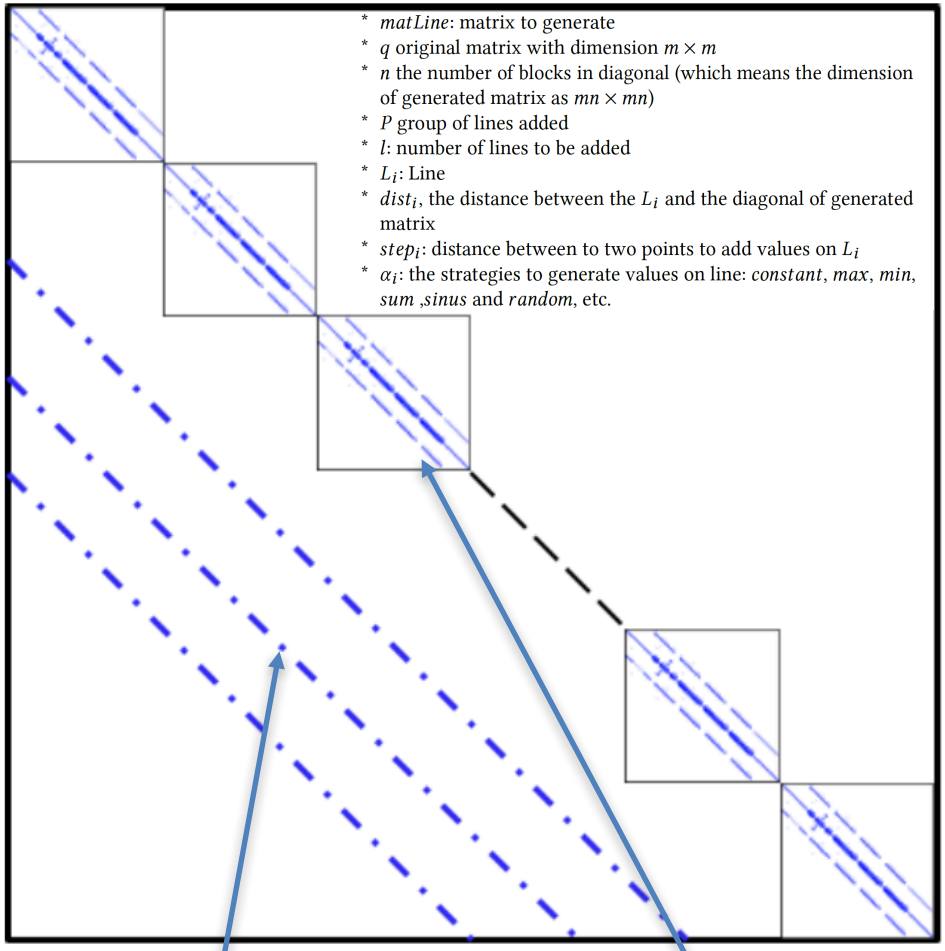
Since experimentations on a platform using asynchronously a Maspar machine and a farm of processors, 1996, U. Lille :

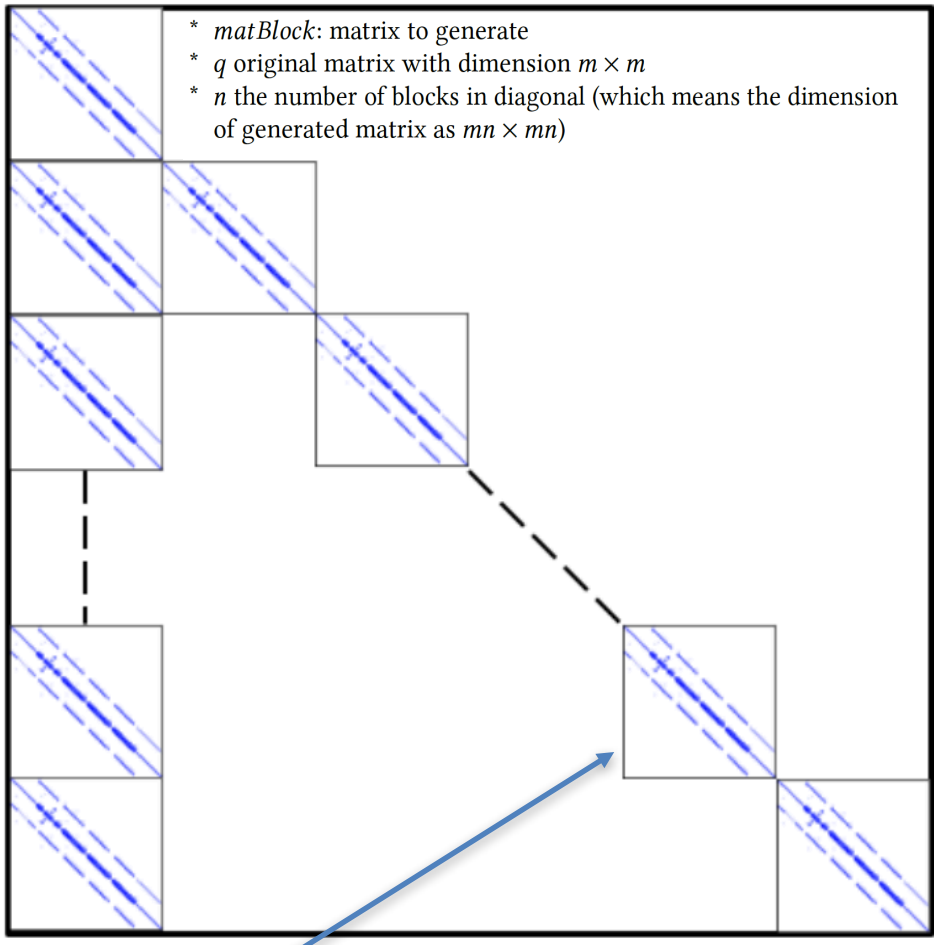


We also experimented on IBM SP and others machines such as the SGI TOTAL supercomputer

We experiment on larger matrices, with an extra level of parallelism (GPU), and with a sequence of linear system, and with faults ,using libraries only.

Sparse Matrix Generator (SMG@mdls) : *matLine* and *matBlock*





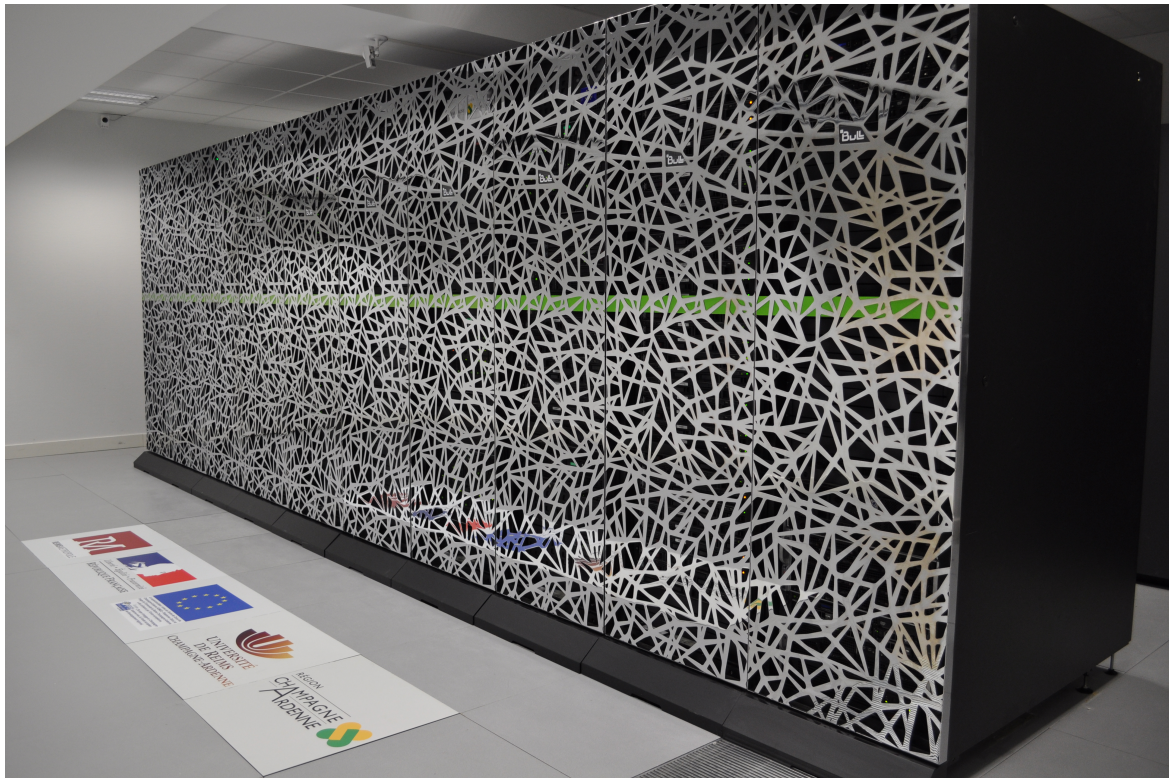
Added Bands

utm300 from matrix market

$$matLine(q, n, P(\bigcup_{i=1}^{l} L_i(dist_i, step_i, \alpha_i)))$$

$$matBlock(q, n)$$

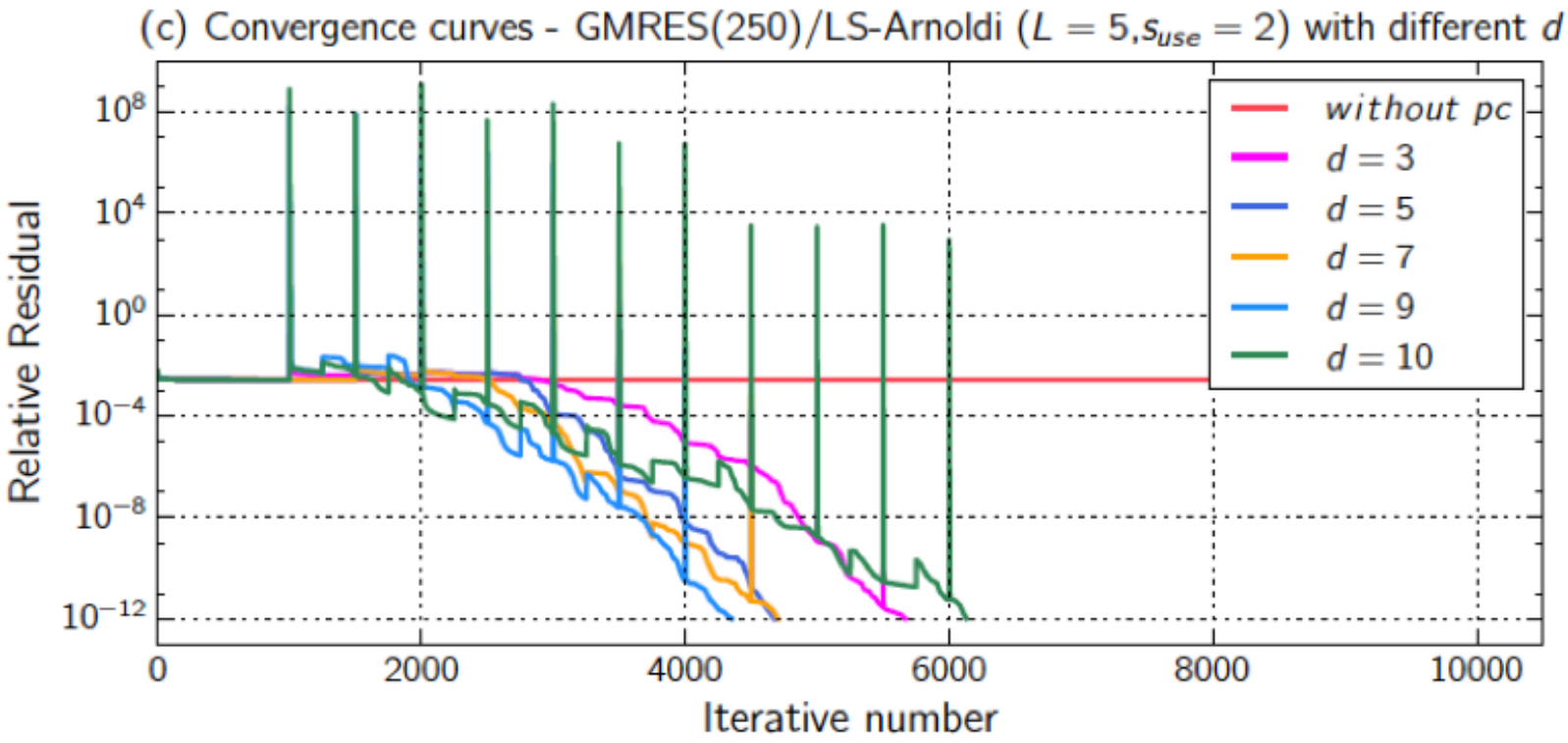
Experiments on the ROMEO Supercomputer in Reims (Champagne, France)



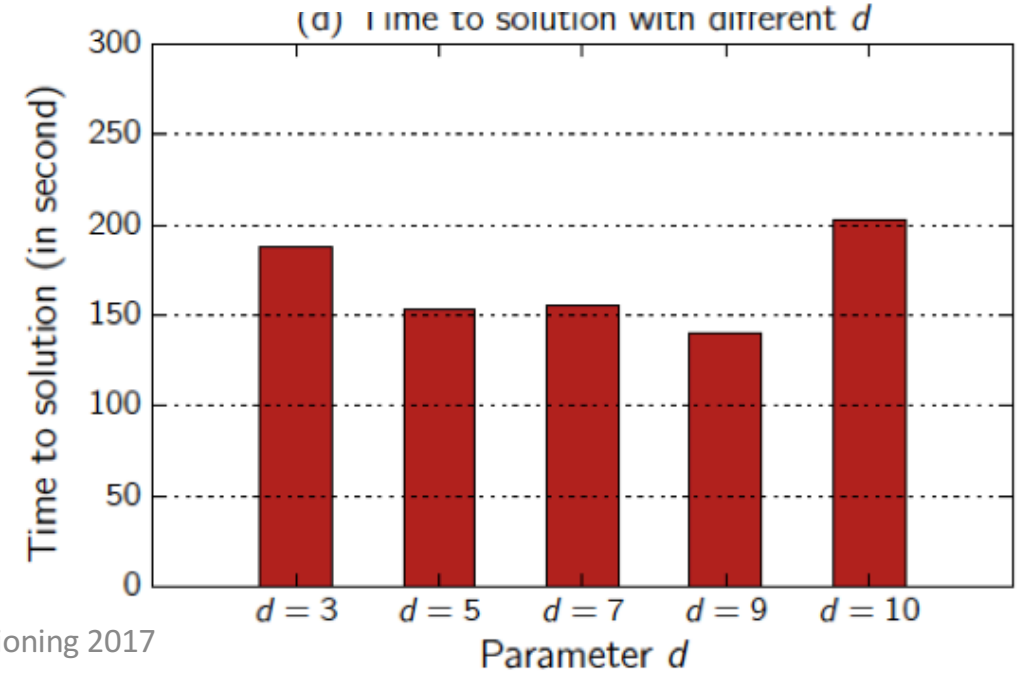
Nodes Number	BullX R421 × 130
Mother Board	SuperMicro X9DRG-QF
CPU	Intel Ivy Bridge 8 cores 2,6 GHz × 2 sockets
Memory	DDR 32GB
GPU	NVIDIA Tesla K20X × 2
Memory	GDDR5 6 GB / GPU

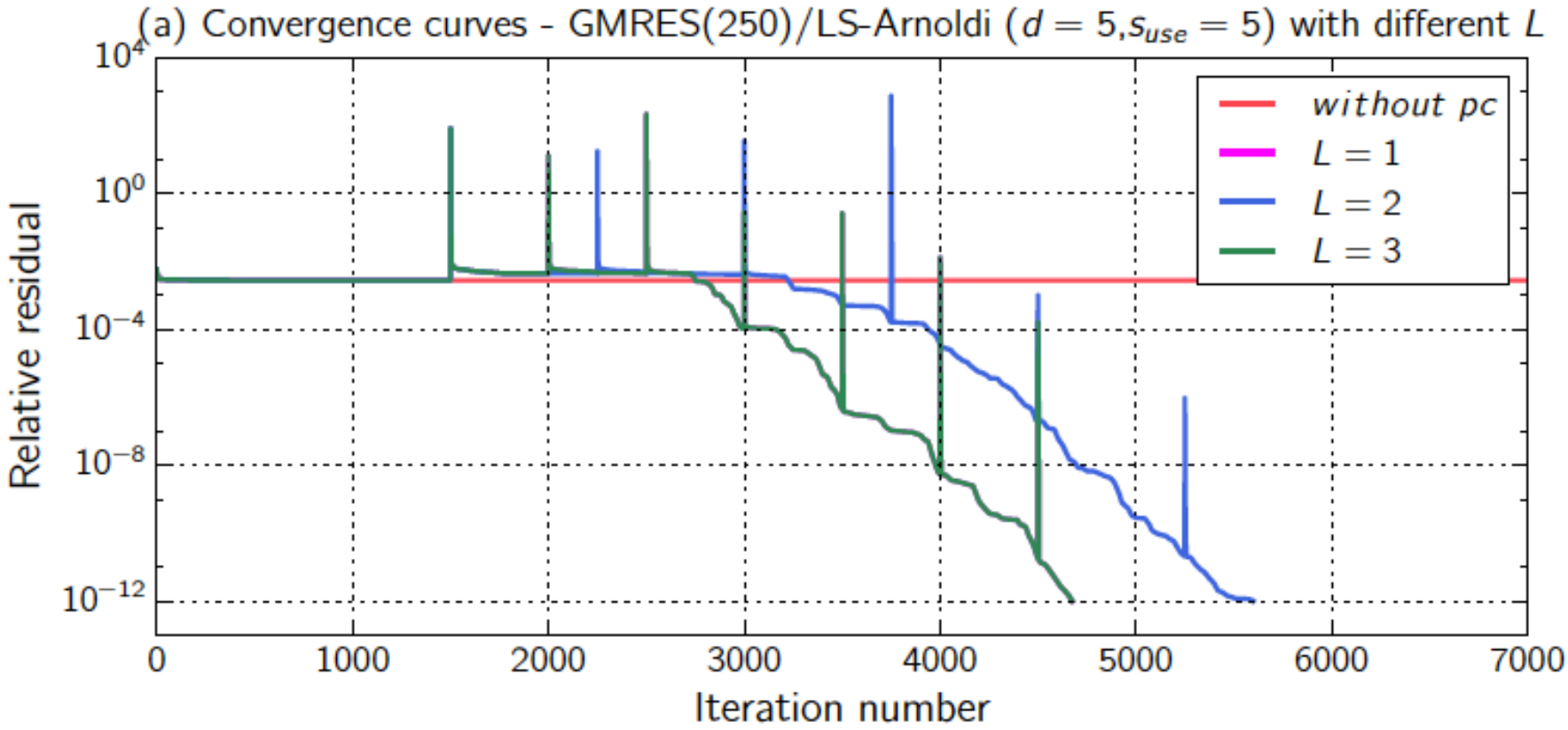
130 nodes, each node has 2 cpu
Each cpu has 8 cores, ie.
16 cores and 2 GPU per node
Infiniband QDR, 40 Gbits/sec

This work was partially supported by the HPC Center of Champagne-Ardenne ROMEO

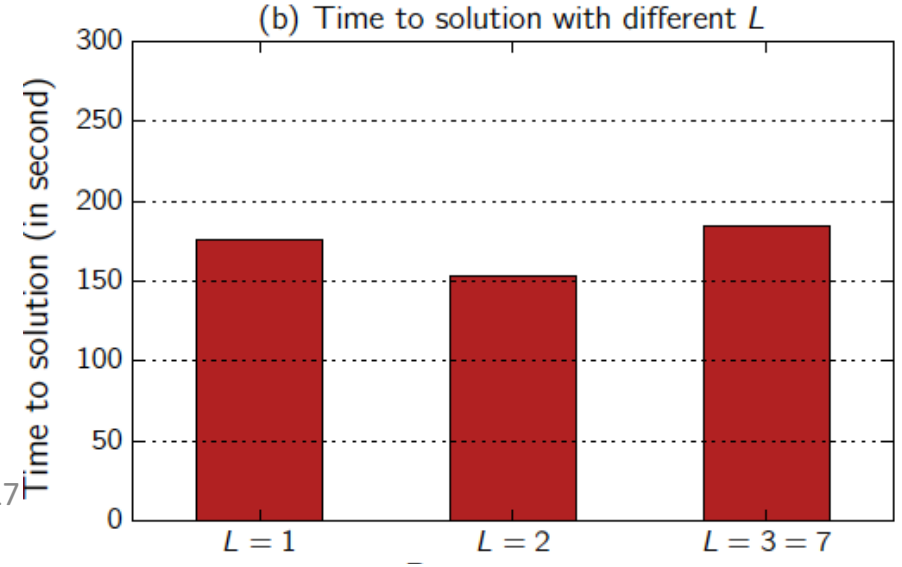


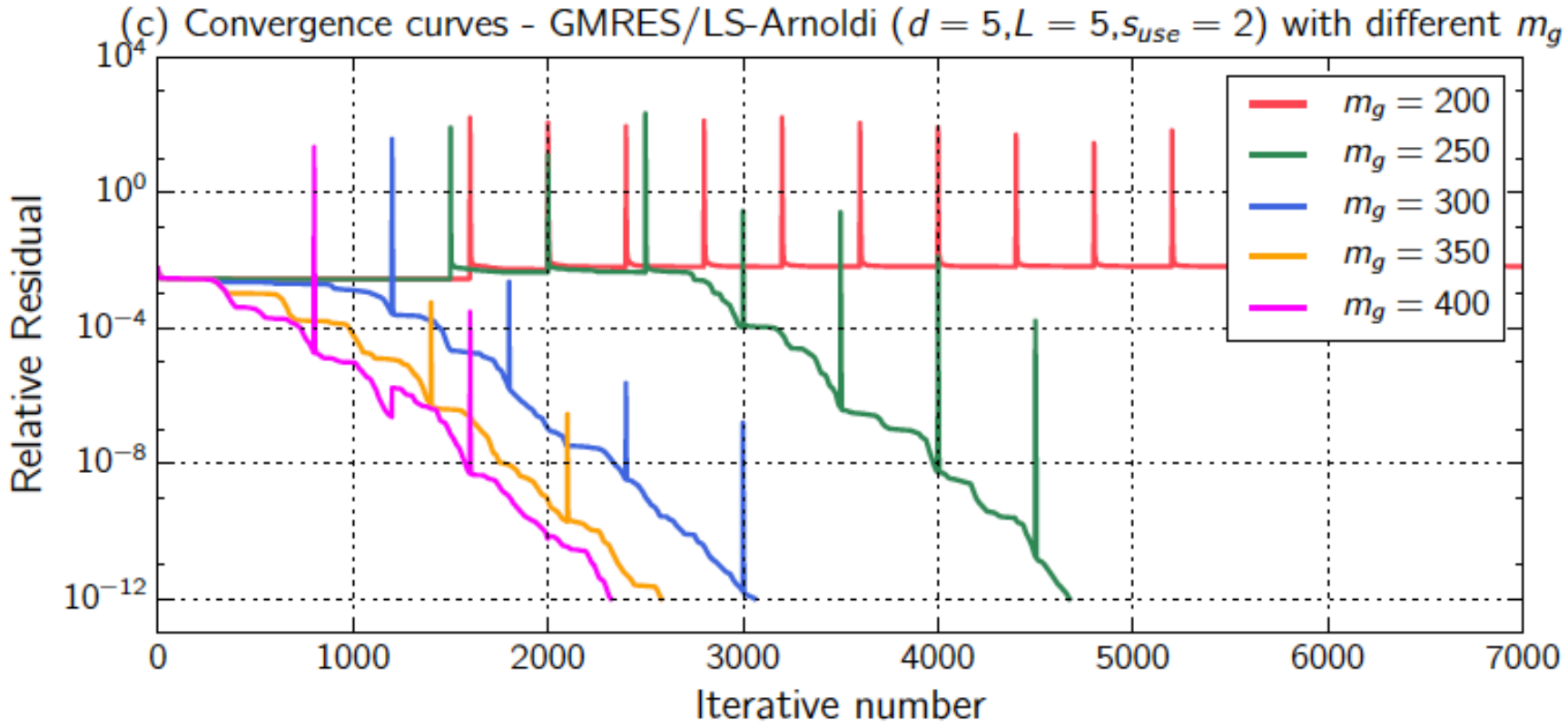
d : degree of the lest square polynomial



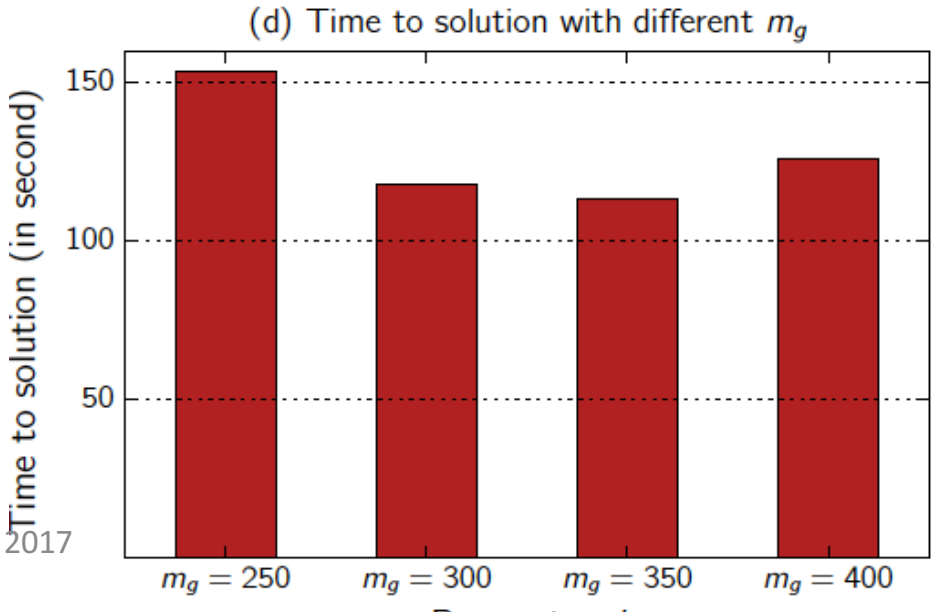


L : number of GMRES restarts between two LS preconditioning





m_g : GMRES subspace size



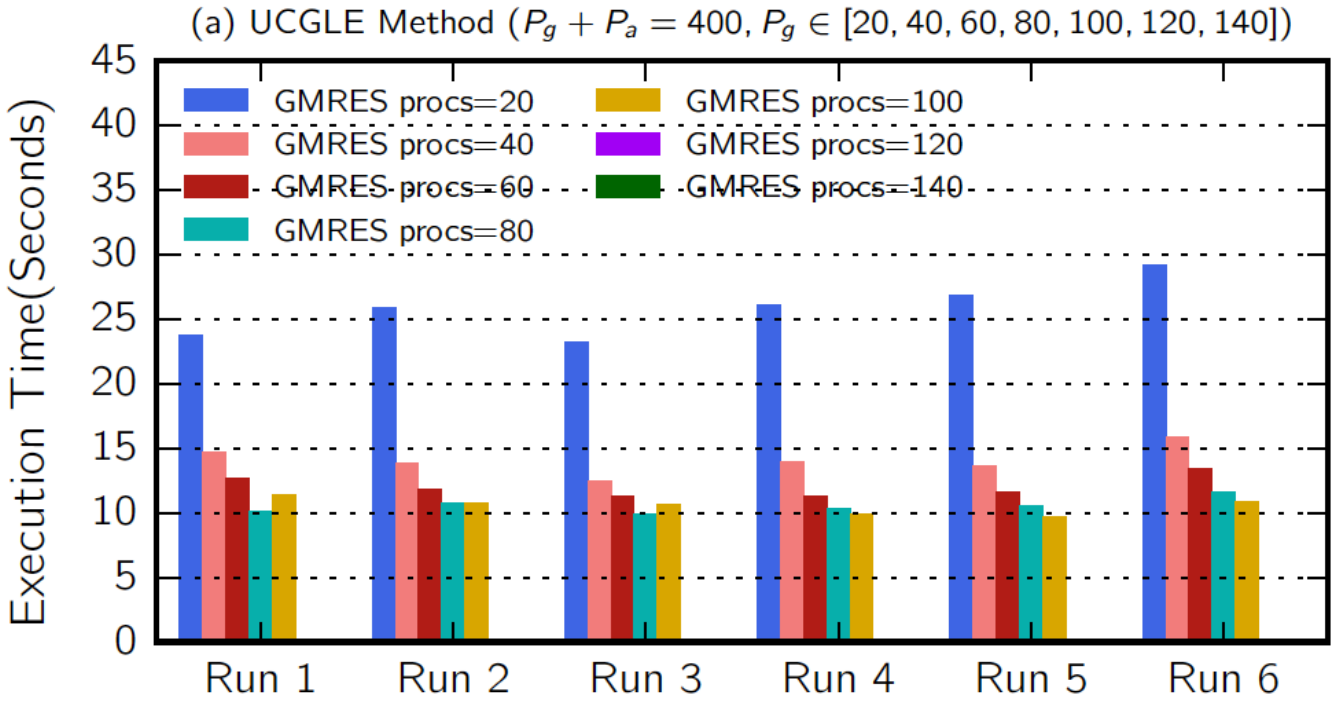
Resolution with different right-hand sides

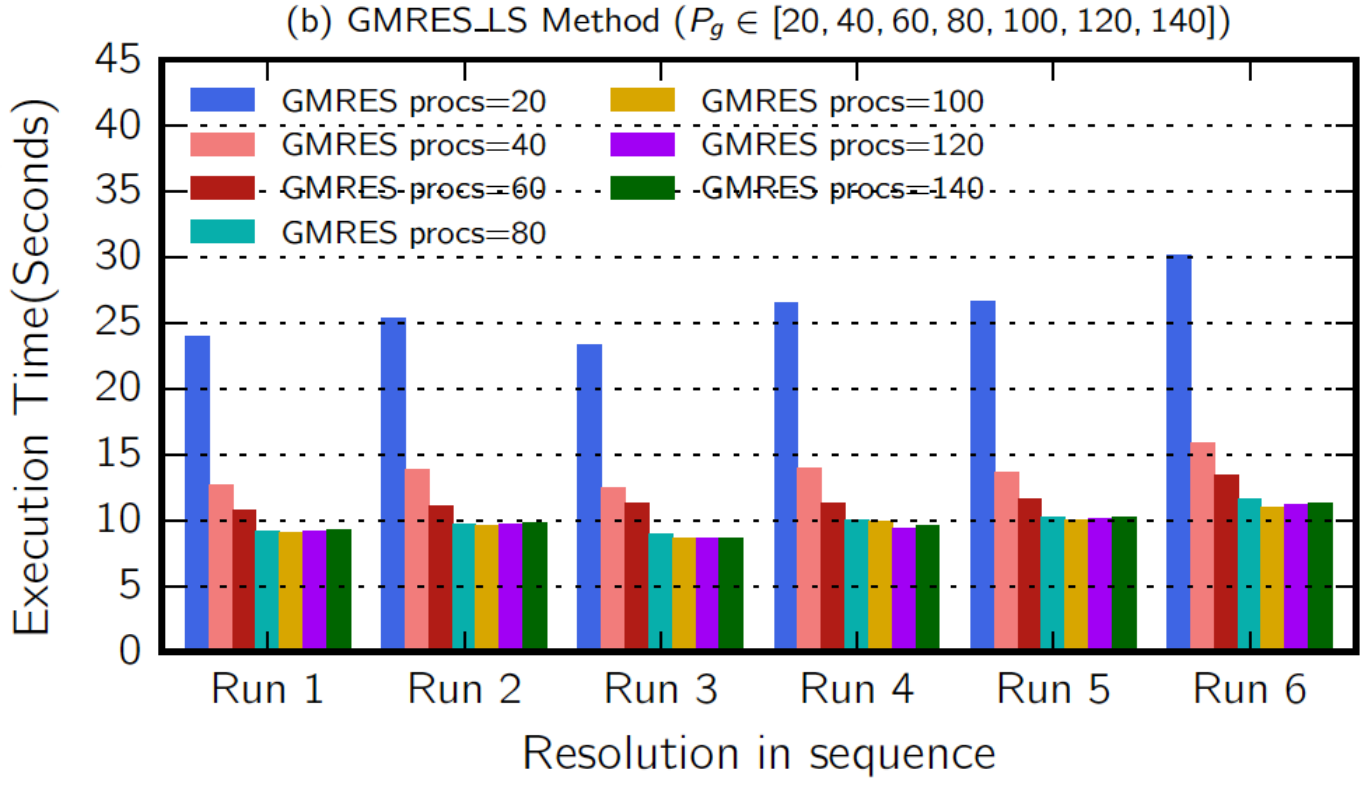
matLine 180K x 180K

$m_g = 220$
$S_{use} = 5$
L =10
d=5

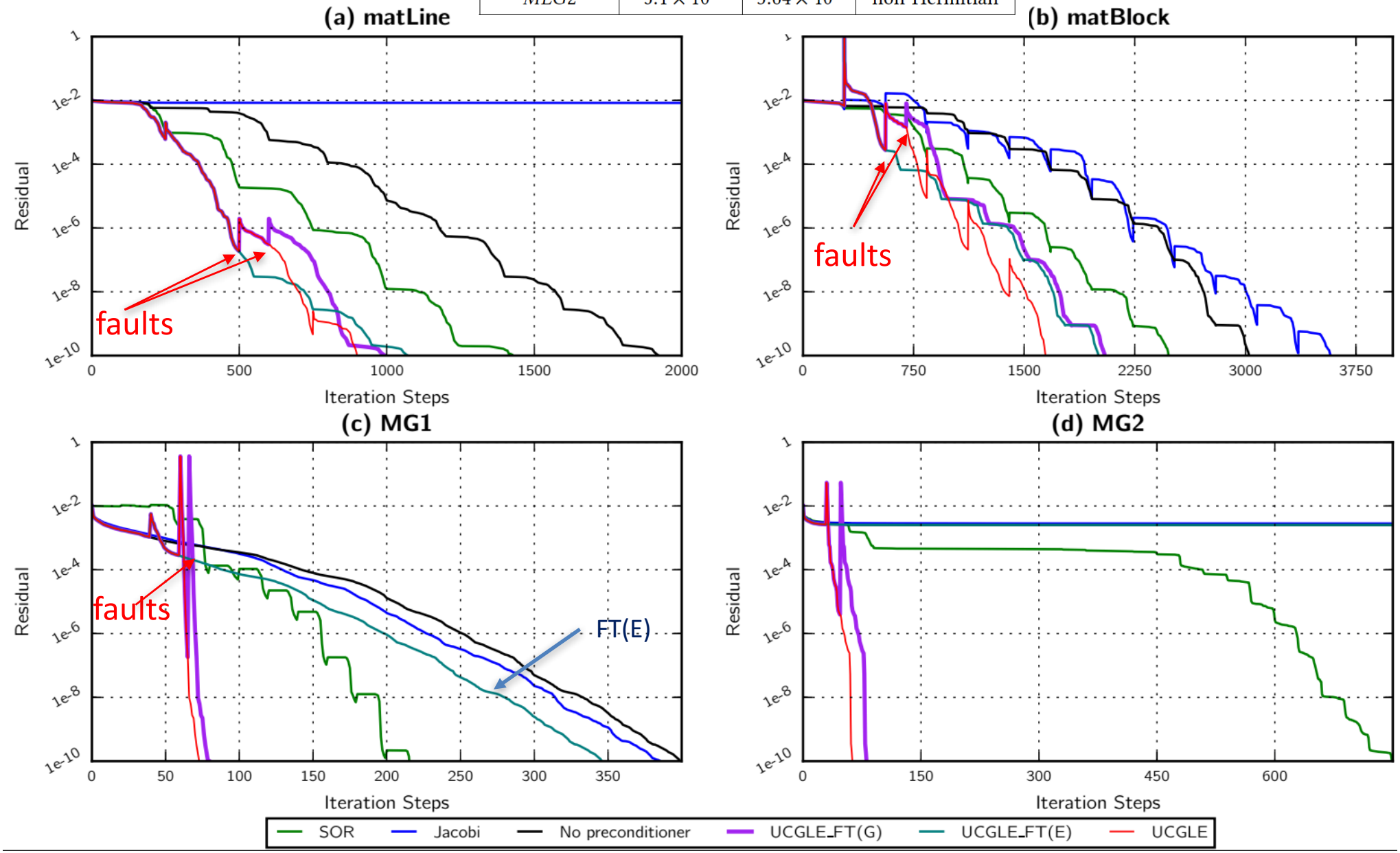
Pa + Pg = 400

Eigenvalues computed before, using UCGLE



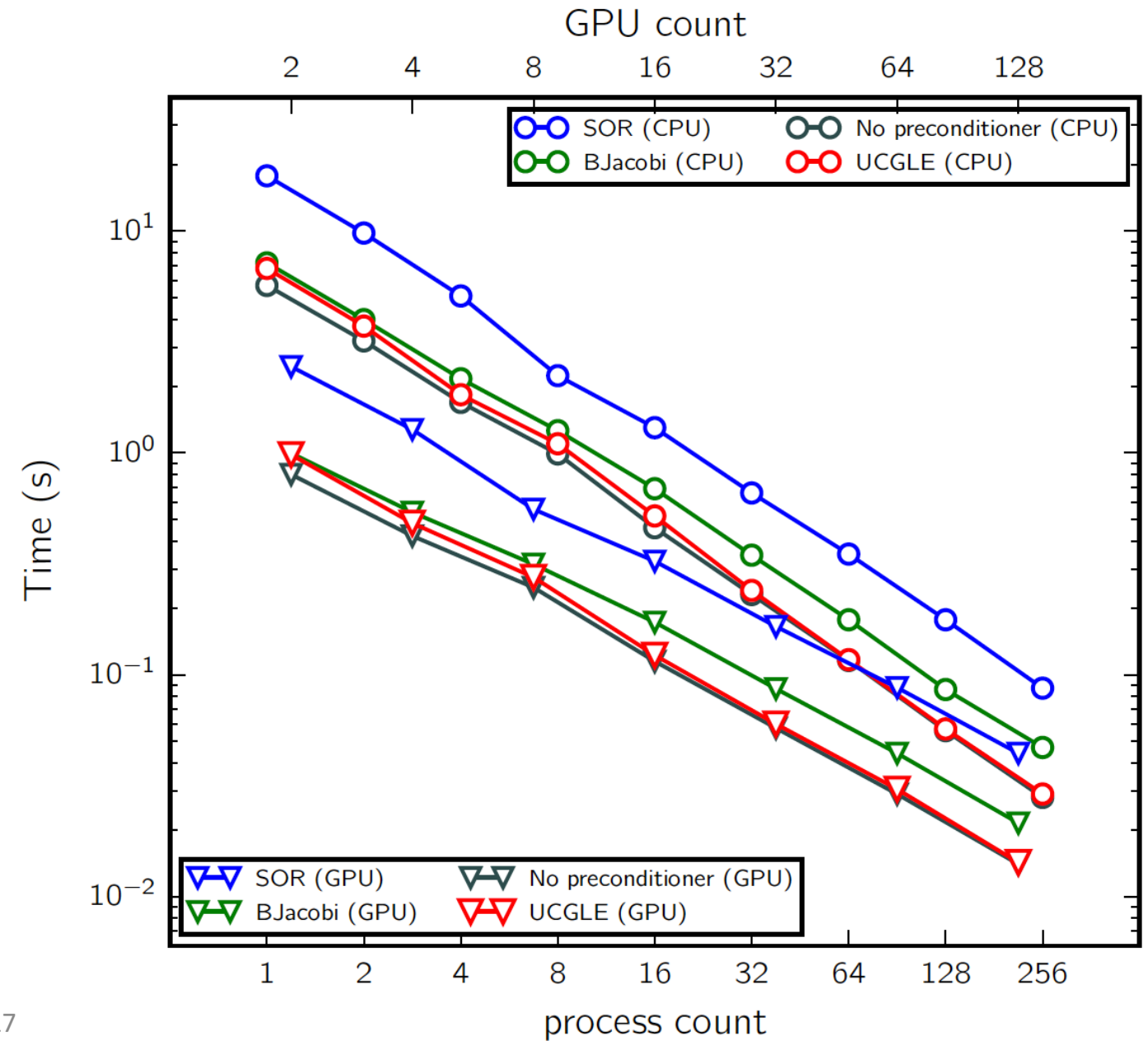


Matrix Name	n	nnz	Matrix Tyoe
matLine	1.8×10^7	2.9×10^7	non-Symmetric
matBlock	1.8×10^7	1.9×10^8	non-Symmetric
MEG1	1.024×10^7	7.27×10^9	non-Hermitian
MEG2	5.1×10^6	3.64×10^9	non-Hermitian

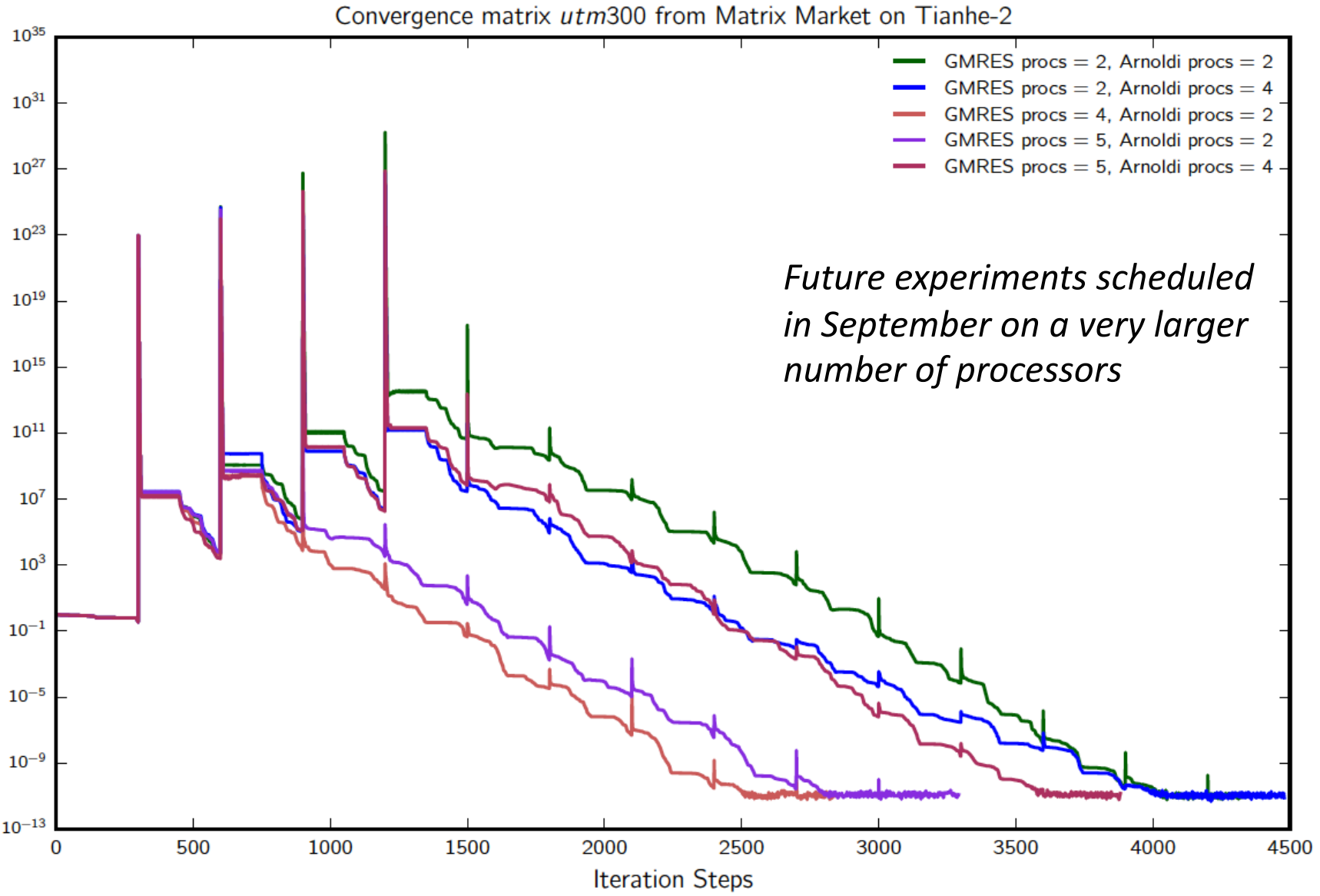


Using PETS and SLEPC

MG1 matrix



First Experiments on Tianhe 2, GZ, China



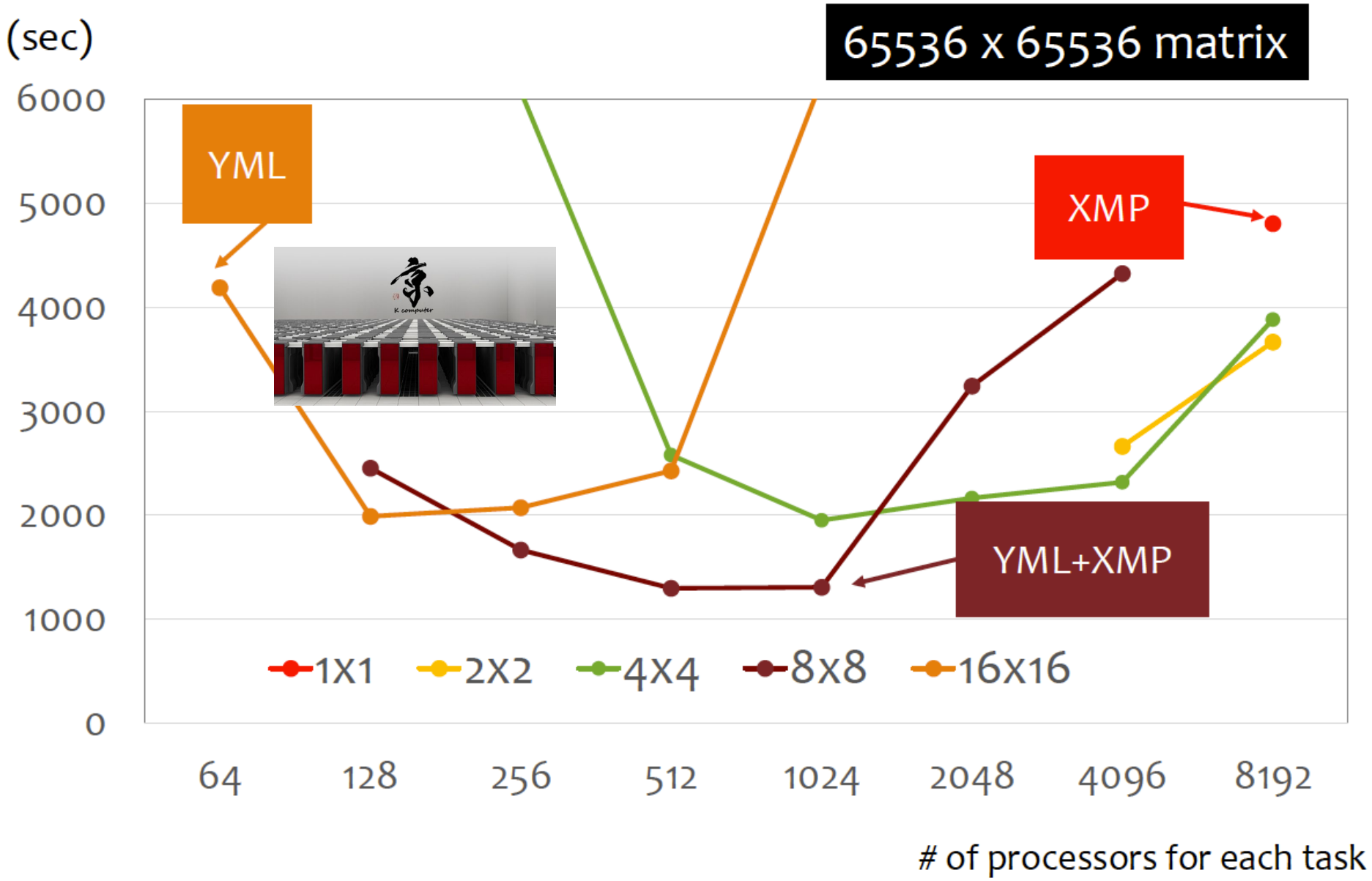
Outline

- Introduction, toward Exascale Computation and beyond
- Distributed and Parallel Programming for Extreme Computing
- The Asynchronous Unite and Conquer GMRES/LS-ERAM (UCGLE) method
- Experimentations, Evaluation and Analysis
- **Conclusion and perspectives**

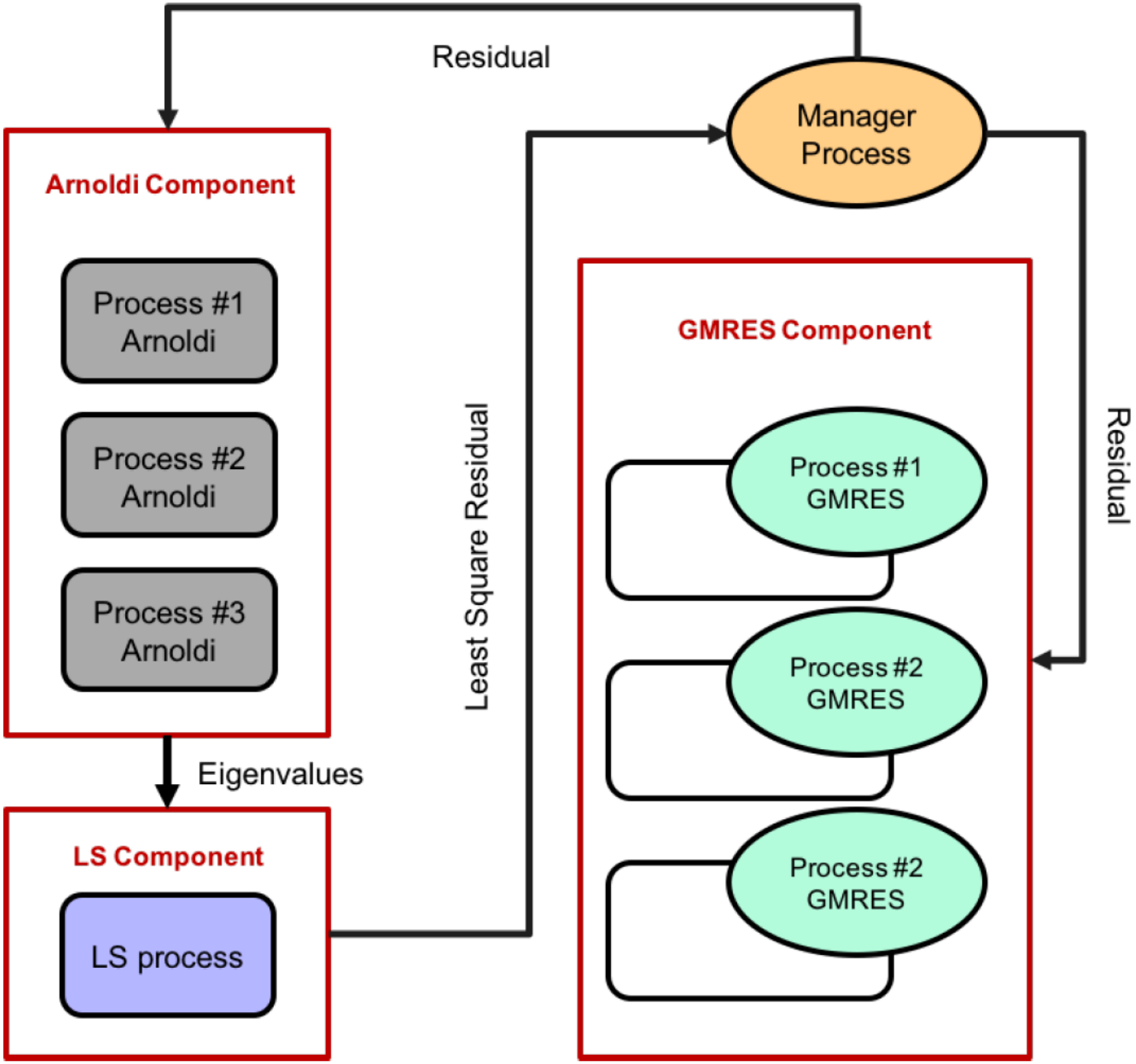
Conclusion and perspectives

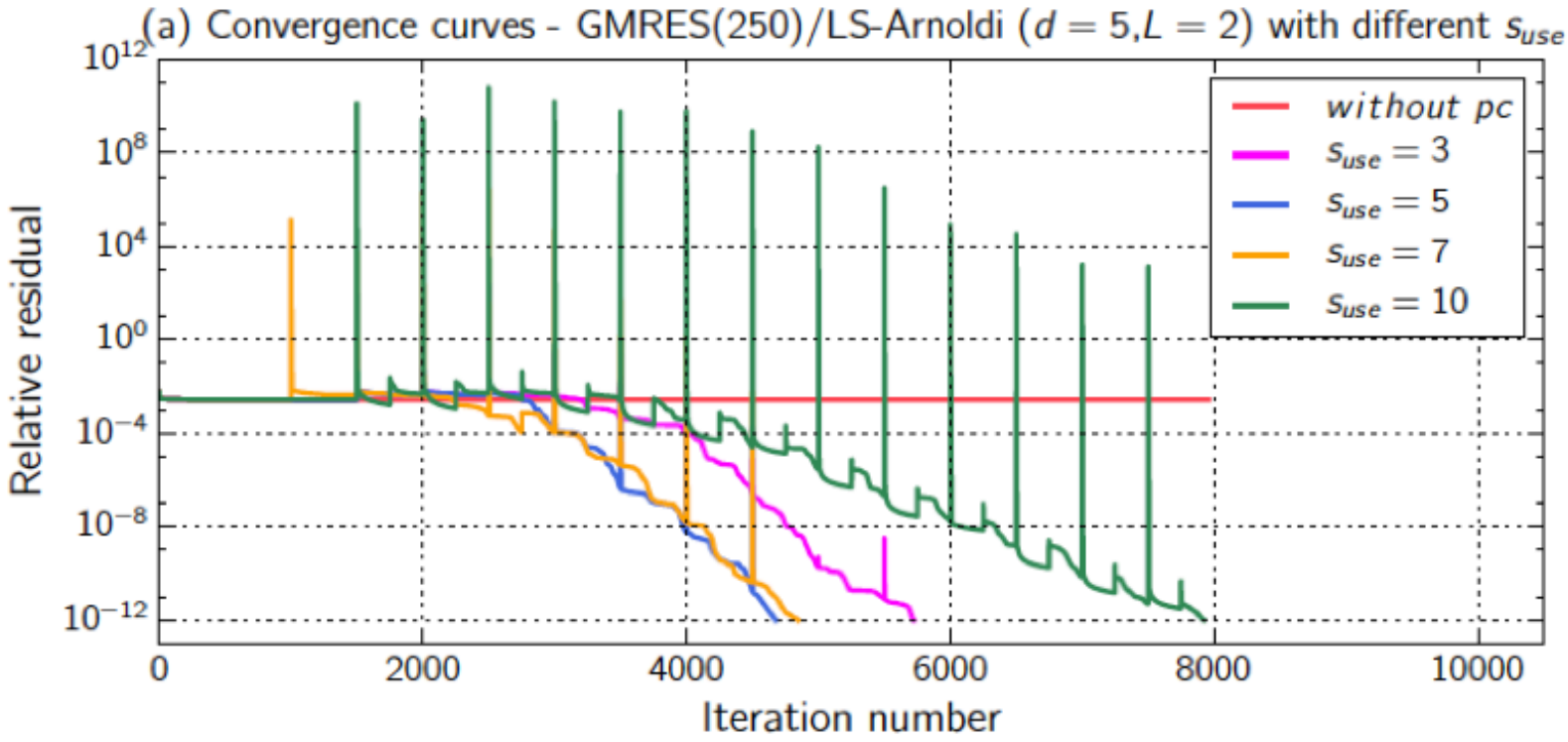
- UCGLE is an asynchronous preconditoned method, minimizing communications, fault tolerant, and allowing efficient GMRES/LS computation for other systems with different right hand sides.
- Several other preconditioners “may be used” (FGMRES) between LS polynomial “accelrations”?
- A lot of parameters have to be analyzed : smart-tuning at runtime, learning, toward intelligent linear algebra
- We have to experiment with very large matrices and on world larger supercomputers to evaluate the impact of large latence for reduction
- Adapted programming paradigms have to be used for such asynchronous multi-granularity distributed and parallel computing (YML-XMP/YML-XACC,…)
- SMG : generation of Non-Hermitian matrices, including some generate with a given spectrum (soon proposed on line)

Experiments (2) BGJ on K-Computer



Slide written by Miwako TSUJI, RIKEN/AICS


Residual
Manager Process
Arnoldi Component
Process #1 Arnoldi
Process #2 Arnoldi
Process #3 Arnoldi
Eigenvalues
LS Component
LS process
Least Square Residual
GMRES Component
Process #1 GMRES
Process #2 GMRES
Process #2 GMRES
Residual



s_{use}:

The number of polynomial acceleration before using "news" received eigenvalues