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

Xinzhe WU[1,2] Serge G. Petiton[1,2]

[1]Maison de la Simulation/CNRS, Gif-sur-Yvette, 91191, France

[2]CRIStAL, University of Lille 1, Science and Technology

January 29, 2018

HPC Asia 2018, Tokyo, Japan

Outline

1. Introduction, toward extreme computing

2. Asynchronous Unite and Conquer GMRES/LS-ERAM (UCGLE) method

3. Experimentations, evaluation and analysis

4. Conclusion and Perspectives

Krylov Methods

Krylov Subspace

$$K_m = span\{r_0, Ar_0, \cdots, A^{m-1}r_0\}$$

Different Krylov Methods:

1. Resolution of linear systems
   - GMRES
   - CG
   - BiCG, etc.
2. Resolution of eigenvalue problems
   - ERAM
   - IRAM, etc.

Future Parallel Programming Trends

Future Programming Trends:

1. Highly hierarchical architectures
   - Computing
   - Memory
2. Increasing levels and degree of parallelism
3. Heterogeneity
   - Computing
   - Memory
   - Scalability
4. Requirement of parallel programming
   - Multi-grain
   - Multi-level memory
   - Reducing synchronizations and promoting asynchronicity
   - Multi-level scheduling strategies

Toward extreme computing, some correlated goals

- Minimize the global computing time
- Accelerate the convergence
- Minimize the number of communications
- Minimize the number of longer size scalar products and reductions
- Minimize the 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

Preconditioning

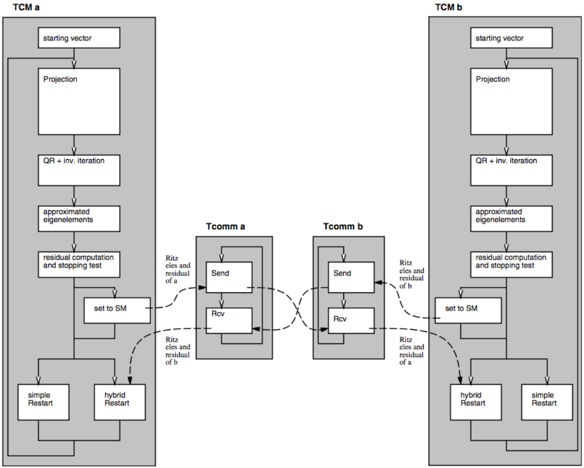
- Minimize the number of communications
- Minimize the number of longer size scalar products and reductions
- Minimize the memory space, cache optimization
- Select the best sparse matrix compressed format
- Mixed arithmetic
- Minimize energy consumption
- Fault tolerance, resilience

Unite and Conquer

Unite and Conquer Approach

Unite and conquer approach: improving the convergence using other iterative methods[Emad, Nahid and Petiton, Serge, 2016].

Figure: Multiple Explicitly Restarted Arnoldi Method (MERAM) [Nahid Emad et al, 2005].

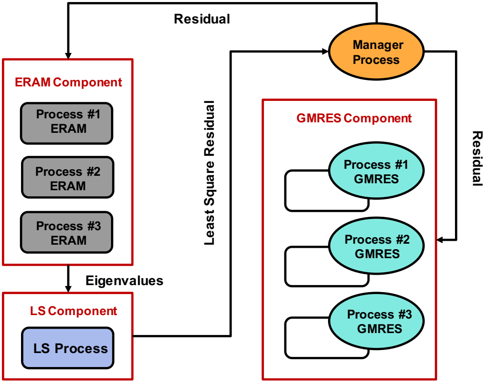


Outline

UCGLE Method Implementation

UCGLE method is proposed to solve the non-Hermitian linear systems based on the work of this article [Essai, Azeddine and Bergére, Guy and Petiton, Serge G, 1999].

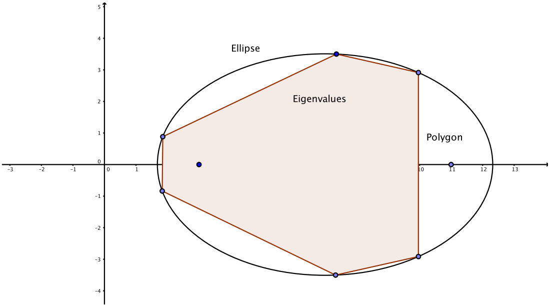
Figure: Workflow of UCGLE method

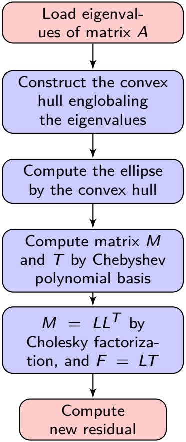


Least Squares Method

Polynomial preconditioner iterates: $x_n = x_0 + P_n(A)r_0 \to r_n = R_n(A)r_0$ 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].

Figure: Eigenvalues, convex hull and ellipse





Least Squares Method

Least Squares method residual

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

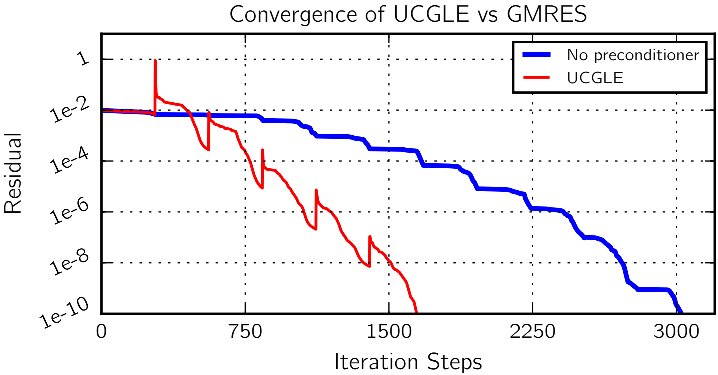
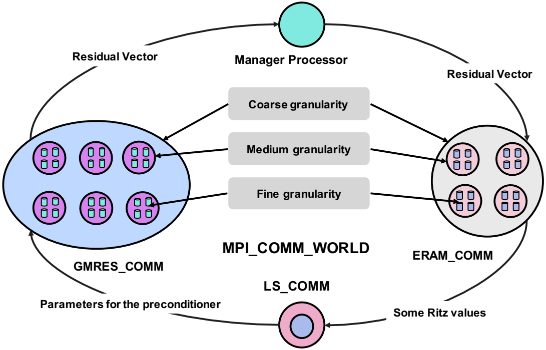


Figure: An example of UCGLE for convergence.

Software engine, orchestration of UCGLE

All three computation components are implemented using the scientific libraries PETSc and SLEPc, based on the work of Pierre-Yves Aquilanti during his thesis at University of Lille 1 [Pierre-Yves Aquilanti, 2011].

Figure: Asynchronous Communication and Parallelism of UCGLE method



Components Implementation

The method implementation is based on the work of Pierre-Yves Aquilanti during his thesis at University of Lille 1 [Pierre-Yves Aquilanti, 2011].

GMRES Component

The GMRES component is well implemented by the PETSc library.

Arnoldi Component

The Arnoldi component is implemented by the SLEPc library to calculate the eigenvalues of the matrix operator A.

LS Component

Using the Cholesky algorithm, which is provided by PETSc as a preconditioner, but can be used without problem as a factorization method correctly.

Important Parameters

There are large number of parameters in UCGLE for the users to select and autotune in order to get the best performance.

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

Outline

Test Matrices

All the following results come from this article [Xinzhe WU and Serge G. Petiton, 2017].

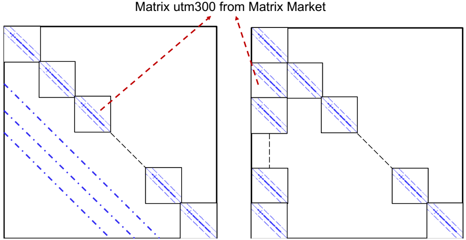


Figure: Two strategies of large and sparse matrix generator

Table: Test matrices information

Matrix Name	n	nnz	Matrix Type
matLine	1.8×10^7	2.9×10^7	non-Symmetric
matBlock	1.8×10^7	1.9×10^8	non-Symmetric
*MEG*1	1.024×10^7	7.27×10^9	non-Hermitian
*MEG*2	5.1×10^6	3.64×10^9	non-Hermitian

Experimental Hardware

Experiments on the ROMEO supercomputer in Reims (Champagne, France). ROMEO has 130 nodes. each node has 2 CPU with 8 cores and 2 GPUs. The node specfication is given as following:

Table: Node Specifications of the cluster ROMEO

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

HPC Center ROMEO Centre de Calcul Régional

Convergence and Fault Tolerance Evaluation

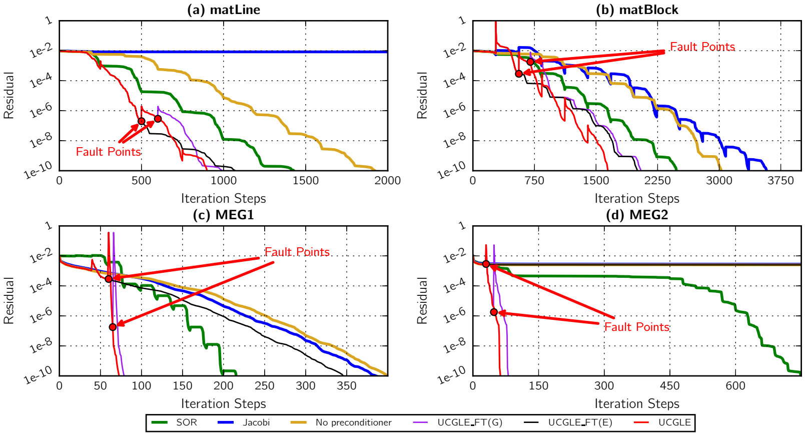


Figure: Convergence comparison of *matLine*, *matBlock*, *MEG*1 and *MEG*2 by UCGLE, classic GMRES, Jacobi preconditoned GMRES, SOR preconditioned GMRES, UCGLE_FT(G) and UCGLE_FT(E); X-axis refers to the iteration step for each method; Y-axis refers to the residual, a base 10 logarithmic scale is used for Y-axis.

Summary of Iteration Number for Convergence

Table: Summary of iteration number for convergence of 4 test matrices using SOR, Jacobi, non preconditioned GMRES,UCGLE_FT(G),UCGLE_FT(G) and UCGLE: red $\times$ in the table presents this solving procedure cannot converge to accurate solution (here absolute residual tolerance 1×10^{-10} for GMRES convergence test) in acceptable iteration number (20000 here).

Matrix Name	SOR	Jacobi	No preconditioner	UCGLE_FT(G)	UCGLE_FT(G)	UCGLE
matLine	1430	$\times$	1924	995	1073	900
matBlock	2481	3579	3027	2048	2005	1646
*MEG*1	217	386	400	81	347	74
*MEG*2	750	$\times$	$\times$	82	$\times$	64

Strong Scalability Results

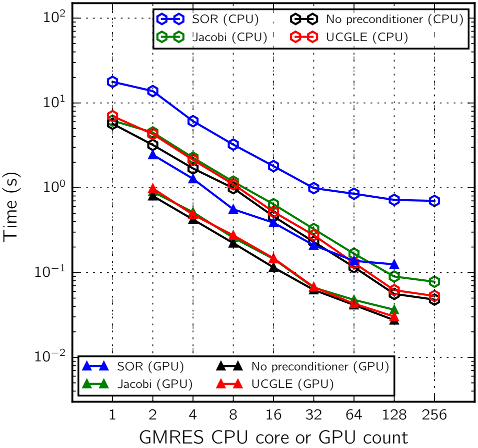


Figure: Strong scalability test of solve time per iteration for UCGLE, GMRES without preconditioner, Jacobi and SOR preconditioned GMRES using matrix *MEG*1 on CPU and GPU; X-axis refers respectively to CPU cores of GMRES from 1 to 256 and GPU number of GMRES from 2 to 128; Y-axis refers to the average execution time per iteration. A base 2 logarithmic scale is used for X-axis, and a base 10 logarithmic scale is used for Y-axis.

Performance Evaluation

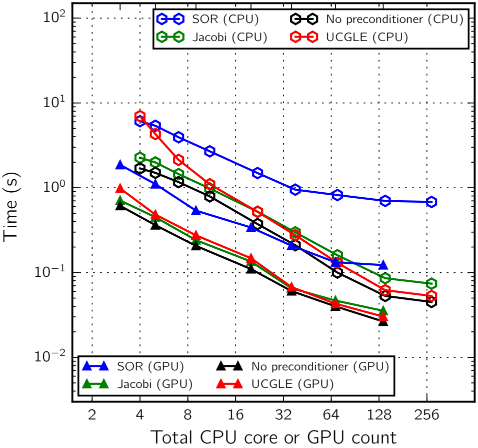


Figure: Performance comparison of solve time per iteration for UCGLE, GMRES without preconditioner, Jacobi and SOR preconditioned GMRES using matrix *MEG*1 on CPU and GPU; X-axis refers respectively to the total CPU cores number or GPU number for these four methods; Y-axis refers to the average execution time per iteration. A base 2 logarithmic scale is used for X-axis, and a base 10 logarithmic scale is used for Y-axis.

Outline

Conclusion and Perspectives

Then

1. UCGLE is an asynchronous preconditoned method, minimizing communications, fault tolerant, and allowing efficient GMRES/LS computation for other systems with different right hand sides;
2. Several other preconditioners "may be used" (FGMRES) between LS polynomial "accelrations";
3. A lot of parameters have to be analyzed : smart-tuning at runtime, learning, toward intelligent linear algebra;
4. We have to experiment with very large matrices and on world larger supercomputers to evaluate the impact of large latence for reduction;
5. Adapted programming paradigms have to be used for such asynchronous multi-granularity distributed and parallel computing (YML-XMP/YML-XACC, $\cdots$);
6. SMG2S: generation of Non-Hermitian matrices, including some generate with a given spectrum (soon proposed on line)

References

Nahid Emad and Serge Petiton (2016)

Unite and conquer approach for high scale numerical computing

Journal of Computational Science, 5 – 14.

Nahid Emad and Serge Petiton and Guy Edjlali (2005)

Multiple explicitly restarted Arnoldi method for solving large eigenproblems

SIAM Journal on Scientific Computing, 253 – 277.

Xinzhe WU, Serge G. Petiton (2018)

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

International Conference on High Performance Computing in Asia-Pacific Region, accepted.

Azeddine Essai, Guy Bergére and Serge G. Petiton (1999)

Heterogeneous Parallel Hybrid GMRES/LS-Arnoldi Method.

PPSC, 1999.

Youssef Saad (1987)

Least squares polynomials in the complex plane and their use for solving nonsymmetric linear systems

SIAM Journal on Numerical Analysis, 155 – 169.

Pierre-Yves Aquilanti (2011)

Methodes de Krylov reparties et massivement paralleles pour la resolution de problemes de Geoscience sur machines heterogenes depassant le petaflop.

PhD dissertation, Universite de Lille.

Thank you for your attentions!

Questions?