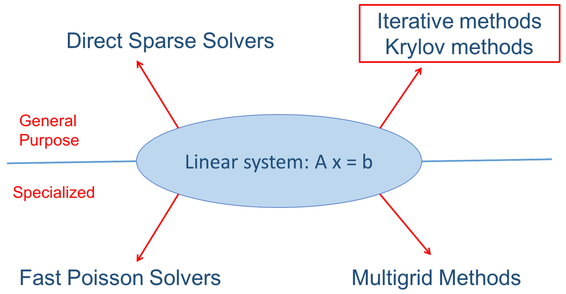
A Distributed and Parallel Asynchronous Unite and Conquer Method to Solve Non-Hermitian Linear Systems

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

[1]Maison de La Simulation, CNRS UMR 3441, 91191, Gif sur Yvette, France
[2]Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL), CNRS UMR 9189, Université de Lille 1, Sciences et Technologies, 59655 Villeneuve d'Ascq Cedex, France

Background

- Linear Systems Solvers



- SuperComputing and Parallel Programming Trends
  - Highly hierarchical architectures (Computing and Memeory)
  - Increasing levels and degree of parallelism
  - Heterogeneity (Computing, Memory and Scalability)
- Requirement of parallel programming
  - Multi-grain, multi-level memory
  - Reducing synchronizations and promoting asynchronicity
  - Multi-level scheduling strategies

Krylov Subspace Methods

Krylov basis computation

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

- Non-Hermitian Linear System Slovers
  - Restarted GMRES
  - Deflated GMRES
  - Flexible GMRES
- Non-Hermitian Eigenvalues Problems Solvers
  - Explicitly Restarted Arnoldi Method (ERAM)
  - Implicitly Restarted Arnoldi Method (IRAM)

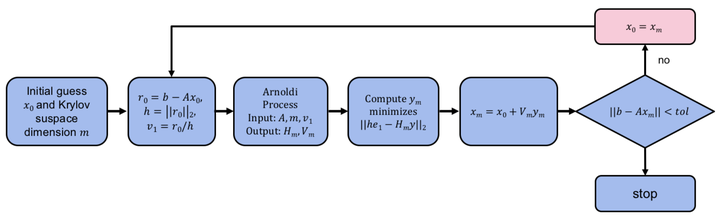


Figure 1: Basic Restarted GMRES Method.

? The improvement is intrinsic to the method. How to design a method which allows the improvement from one time resolution to another?

Least Squares Polynomial Method

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 [2].

Least Squares method residual

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

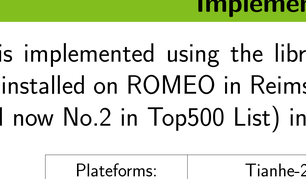


Figure 2: Eigenvalues, convex hull and ellipse.

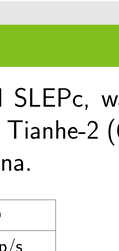


Figure 3: Least Squares Polynomial Preconditioner.

UCGLE Method

UCGLE: Unite and Conquer [1] GMRES-LS/ERAM method

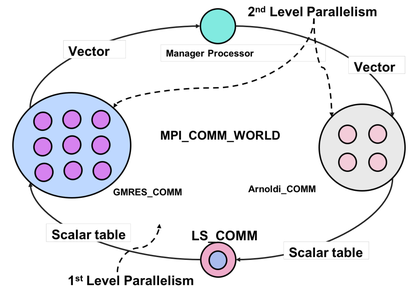


Figure 4: UCGLE Method Workflow

Implementation

UCGLE is implemented using the libraries PETSc and SLEPc, was successfully installed on ROMEO in Reims, France, and on Tianhe-2 (6 times No.1 and now No.2 in Top500 List) in Guangzhou, China.

Plateforms:	Tianhe-2	ROMEO
Speed (LINPACK)	33,862.7 TFlop/s	254.9 TFlop/s
Power	17,808.00 kW	81.41 kW
Architectures	3,120,000 Intel Xeon E5-2692v2 12C 2.2GHz	5, 720 Intel Xeon E5-2650v2 8C 2.6GHz

Experimental Results

Convergence Comparison and Performance Comparison [3]:

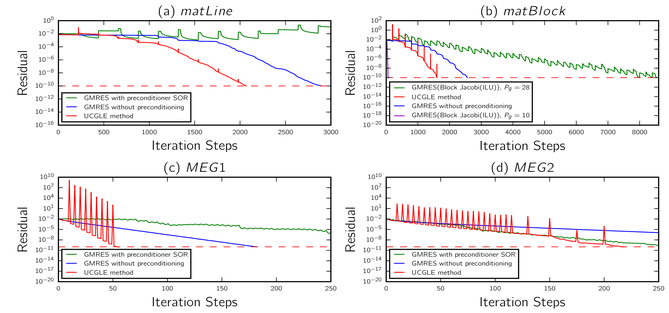


Figure 5: Convergence analysis of UCGLE method, traditional restarted GMRES and GMRES with different preconditioners.

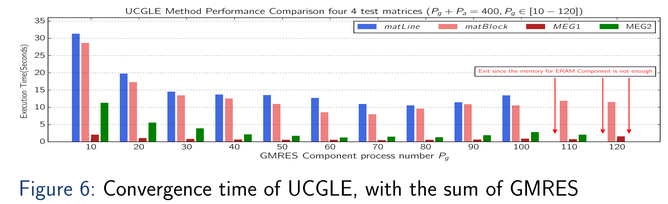


Figure 6: Convergence time of UCGLE, with the sum of GMRES Component process number and ERAM Component process number fixed

Summary and Perspective

This poster outlines a distributed and parallel method UCGLE for the resolution of non-Hermitian linear systems to explore the novel numerical parallel methods.

In future, there are still lots of aspects to evaluate. Firstly, we should study the the impacts of different parameters of each component on the performance of UCGLE. Secondly, it's necessary to study the influence of the eigenvalues on UCGLE, and summarize the scenes suited to this method. In the end, we plan to implement this method based on the workflow and distributed parallel methodologies YML-XMP, a kind of user-friendly and hierarchical-system-oriented development and execution environment.

References

[1] N. Emad and S. Petiton.
Unite and conquer approach for high scale numerical computing.
Journal of Computational Science, 14:5–14, 2016.

[2] Y. 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.

[3] X. Wu and S. G. Petiton.
A distributed and parallel asynchronous unite and conquer method to solve large scale non-hermitian linear systems.
Submitted to IEEE CLUSTER 2017.