A Scalable Generator of Non-Hermitian Test Matrices computed from Given Spectra for Large-scale Systems

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

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

[2]CRIStAL, Université de Lille, France

3rd Workshop on Parallel Programming Models - Productivity and Applications

March 15, 2018

Aachen, Germany

Outline

Eigenvalues and eigenvalue problems

Eigenvalues and eigenvectors

For a square matrix A, if there is a vector $u \in \mathbb{C}^n$ such that

$$Au = \lambda u$$

for some scalar λ, then λ is called the eigenvalue of A with corresponding (right) eigenvector u.

Applications of eigenvalue problems:

1. numerical simulation
   - the Schrödinger equation [8], molecular simulation [11], geology [7], etc.
   - preconditioners for solving linear systems, e.g. UCGLE [12].
2. machine learning and pattern recognition
   - principal component analysis (PCA) [4]
   - Fisher discriminant analysis (FDA) [2]
   - clustering [9], etc.

Requirement of large-scale matrix generator

The backgroud:

- the eigenvalue problem size in both machine learning and numerical simulation is increasing;
- the numerical methods should be ajusted to the coming exascale platforms.

Thus there are three special requirements on the test matrices for the evaluation of numerical algorithms:

- their spectra must be known and can be easily controlled;
- they should be both sparse, non-Hermitian and non-trivial;
- they could have a very high dimension, which includes the non-zero element numbers and/or the matrix dimension to evaluate the algorithms on large-scale systems.

Related works

The related work:

- the Time Davis collection [5];
- the Matrix Market collection [3];
- Bai's collection [1];
- J. Demmel's generation suite in 1989 to benchmark LAPACK [6], etc.

Only the proposed method by J. Demmel generate the test matrices with given spectra, which can transfer the diagonal matrix with given spectra into a dense matrix with same spectra using the orthogonal matrices, and then reduce them to unsymmetric band ones by the Householder transformation. This method requires $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ storage even for generating a small bandwidth matrix.

Outline

Mathematical notations (H. Galicher et. al)

For all matrices $A \in \mathbb{C}^{n\times n}$, $M \in \mathbb{C}^{n\times n}$, $n \in \mathbb{N}$, a linear operator $\widetilde{A_A}$ of matrix M determined by matrix A can be set up as Formule (1):

$$\begin{cases} \widetilde{A_A} : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}, \\ \qquad M \to AM - MA. \end{cases} \tag{1}$$

$$(\widetilde{A_A})^k(M_0) = \sum_{m=0}^{k} (-1)^m C_k^m A^{k-m} M_0 A^m. \tag{2}$$

$$M_{i+1} = M_i + \frac{1}{i!}(\widetilde{A_A})^i(M_0), i \in (0, +\infty). \tag{3}$$

In order to make $\widetilde{(A_A)}^i$ tends to $\mathbf{0}$ in limited steps, it is necessary that $A = B^{-1}PB$, then we set the matrix P to be nilpotent, and the matrix B to be the identity matrix $I \in \mathbb{N}^{n\times n}$ for simplification based on the preliminary theoretical research [10].

SMG2S Algorithm (H. Galicher et. al)

The SMG2S algorithm is given as:

Algorithm 1 Matrix Generation Method

```
Input: $Spec_{in} \in \mathbb{C}^n$, $h$, $d$
Output: $M_t \in \mathbb{C}^{n \times n}$
1: Insert random elements in $h$ lower diagonals of $M_o \in \mathbb{N}^{n \times n}$
2: Insert $Spec_{in}$ on the diagonal of $M_0$ and $M_0 = (2d-2)!M_0$
3: Randomly insert 1 and 0 on sub-diagonal of $A \in \mathbb{N}^{n \times n}$ with the maximum continuous length of 1 to be $d$
4: for $i = 0, \cdots, 2(d-2)-1$ do
5:     $M_{i+1} = M_i + (\prod_{k=i+1}^{2d-2} k)(\widetilde{A_A})^i(M_0)$
6: end for
7: $M_t = \frac{1}{(2d-2)!} M_{2d-2}$
```

Parallel Implementation of CPUs and GPUs (X. Wu and S. Petiton)

We implement SMG2S on homogenous and heterogeneous machines. The former is implemented based on MPI and PETSc[1], the latter is based on MPI, CUDA, and PETSc. The kernel of implementation is the SpGEMM.

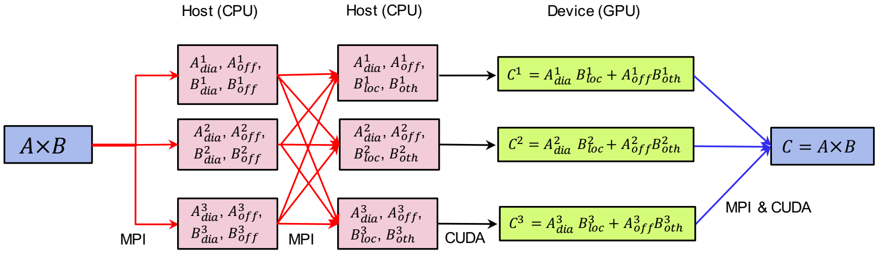

Host (CPU)
Host (CPU)
Device (GPU)
$A \times B$
$A^1_{dia}, A^1_{off}, B^1_{dia}, B^1_{off}$
$A^2_{dia}, A^2_{off}, B^2_{dia}, B^2_{off}$
$A^3_{dia}, A^3_{off}, B^3_{dia}, B^3_{off}$
$A^1_{dia}, A^1_{off}, B^1_{loc}, B^1_{oth}$
$A^2_{dia}, A^2_{off}, B^2_{loc}, B^2_{oth}$
$A^3_{dia}, A^3_{off}, B^3_{loc}, B^3_{oth}$
$C^1 = A^1_{dia} B^1_{loc} + A^1_{off} B^1_{oth}$
$C^2 = A^2_{dia} B^2_{loc} + A^2_{off} B^2_{oth}$
$C^3 = A^3_{dia} B^3_{loc} + A^3_{off} B^3_{oth}$
$C = A \times B$
MPI
MPI
CUDA
MPI & CUDA

Figure: The structure of a CPU-GPU implementation of SpGEMM, where each GPU is attached to a CPU. The GPU is in charge of the computation, while the CPU handles the MPI communication among processes.

[1]Portable, Extensible Toolkit for Scientific Computation

Outline

Experimental hardware environment

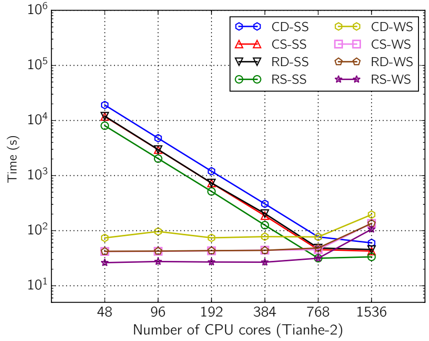
We implement SMG2S on the supercomputers *Tianhe-2* and *Romeo*. The node specfication for the two platforms is given as following:

Table: Node Specifications of the cluster ROMEO and Tianhe-2

Machine Name	ROMEO	Tiahhe-2
Nodes Number	BullX R421 $\times$ 130	16000 $\times$ nodes
Mother Board	SuperMicro X9DRG-QF	Specific Infiniband
CPU	2$\times$Intel Ivy Bridge 8 cores 2.6 GHz	2$\times$Intel Ivy Bridge 12 cores 2.2 GHz
Memory	DDR3 32GB	DDR3 64GB
Accelerator	NVIDIA GPU Tesla K20X $\times$ 2	Intel Knights Corner $\times$ 3

Strong and Weak Scalability Evaluation (X. Wu and S. Petiton)

The strong and weak scaling tests on CPUs are given as:



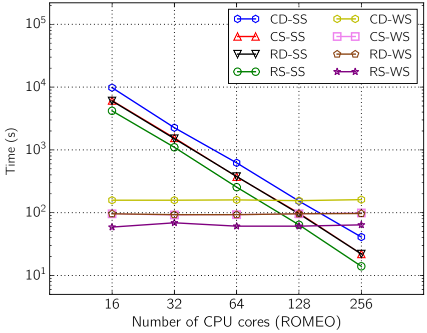


Figure: Strong and weak scalability on *Tianhe-2* and *Romeo*. A base 2 logarithmic scale is used for X-axis, and a base 10 logarithmic scale for Y-axis."CD" is short for "complex double", "CS" for "complex single", "RD" for "real double", "RS" for "real single", "SS" for "strong scalability", and "WS" for "weak scalability". On *Tianhe-2*, the matrix size for strong scalability is 1.6×10^7, and the matrix sizes for weak scalability range from 1.0×10^6 to 3.2×10^7. On *Romeo*, the matrix size for strong scalability is 3.2×10^6, and the matrix sizes for weak scalability range from 4.0×10^5 to 6.4×10^6. h and d are respectively 8 and 4.

Strong and Weak Scalability Evaluation (X. Wu and S. Petiton)

The strong and weak scaling tests on multi-GPUs are given as:

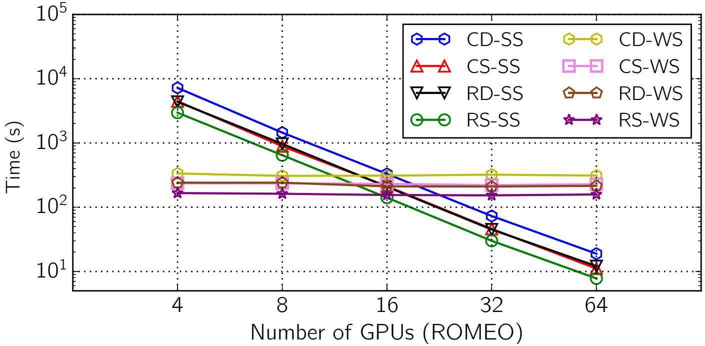


Figure: Strong and weak scalability of GPUs on *Romeo*. A base 2 logarithmic scale is used for X-axis, and a base 10 logarithmic scale for Y-axis."CD" is short for "complex double", "CS" for "complex single", "RD" for "real double", "RS" for "real single", "SS" for "strong scalability", and "WS" for "weak scalability". The matrix size for strong scalability is 8.0×10^5, and the matrix sizes for weak scalability range from 2.0×10^5 to 3.2×10^6. h and d are respectively 8 and 4.

Multi-GPU Speedup Evaluation (X. Wu and S. Petiton)

The multi-GPUs speedup over CPUs is given as:

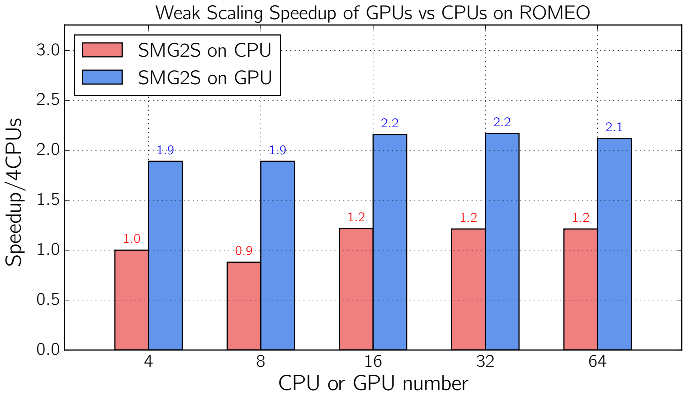


Figure: Weak scaling speedup of GPUs vs CPUs on *Romeo* with real double scalar type. X-axis refers to computing unit number from 4 to 64, and Y-axis refers to the speedup of CPUs or GPUs over time spent by 4 CPUs with matrix size 2.0×10^5. The matrix sizes for the weak scalability are respectively 2.0×10^5, 4.0×10^5, 8.0×10^5, 1.6×10^6 and 3.2×10^6. h and d are respectively 8 and 4.

Outline

Verification method (X. Wu and S. Petiton)

We proposed a method to check the ability of SMG2S to keep the given spectra based on the Shifted Inverse Power Method.

Algorithm 2 Shifted Inverse Power Method

Input: Matrix A, initial guess for desired eigenvalue σ, initial vector v_0
Output: Approximate eigenpair (θ, v)

```
1: $y = v_0$
2: for $i = 1, 2, 3 \cdots$ do
3:     $\theta = ||y||_\infty$, $v = y/\theta$
4:     Solve $(A - \sigma I)y = v$
5: end for
```

Check error

$$error = \frac{||Av' - \lambda v'||}{||Av'||}$$

Verification results (X. Wu and S. Petiton)

The verification tests have been done with 4 different types of spectra.

Table: Test Spectra information

Spectra Name	*spec*1	*spec*2	*spec*3	*spec*4
Scalar Type	complex	real	complex	real
Spectra Interval	[10,1000]	[10,1000]	[5,500]	[5,500]

Verification results (X. Wu and S. Petiton)

The accuracy verification results are given as:

Table: Accuracy Verification Results.

Matrix N^o	Size	Spectra	precision	Accuracy	Acceptance (%)	max *error*
1	100	*spec*1	double	1×10^{-13}	100	6×10^{-14}
2	100	*spec*1	single	1×10^{-6}	100	3×10^{-7}
3	100	*spec*2	double	1×10^{-13}	100	8×10^{-14}
4	100	*spec*2	single	1×10^{-6}	97	3×10^{-3}
5	100	*spec*3	double	1×10^{-15}	100	4×10^{-16}
6	100	*spec*3	single	1×10^{-6}	100	6×10^{-7}
7	100	*spec*4	double	1×10^{-15}	94	4×10^{-4}
8	100	*spec*4	single	1×10^{-6}	100	9×10^{-7}

Outline

Conclusion and Perspectives

Then

1. SMG2S is a method to generate large-scale non-Hermitian matrices with good scalabilities;
2. SMG2S has capacity to keep the accuracy of given spectra;
3. An open source software should be implemented based on the basic C or C++, CUDA and MPI without PETSc and other large libraries;
4. The matrix-matrix multiplication kernel should be optimized and specified for both CPUs and multi-GPUs.

Acknowledgement

- We would like to thank Prof. Yutong LU and their team in the National Supercomputing Center in Guangzhou for providing the use of Tianhe-2.
- This work is partially supported by the ROMEO HPC Center Champagne Ardenne for providing the use of cluster Romeo.
- This work is supported by the German-Japanese-French project MYX partially supported by the French National Research Agency (ANR) under the SPPEXA framework.

References I

[1] Z. Bai, D. Day, J. Demmel, and J. Dongarra.
A test matrix collection for non-hermitian eigenvalue problems.
Prof. Z. Bai, Dept. of Mathematics, 751:40506–0027, 1996.

[2] P. Berkes.
Handwritten digit recognition with nonlinear fisher discriminant analysis.
In *Proceedings of the 15th international conference on Artificial neural networks: formal models and their applications-Volume Part II*, pages 285–287. Springer-Verlag, 2005.

[3] R. F. Boisvert, R. Pozo, K. Remington, R. F. Barrett, and J. J. Dongarra.
Matrix market: a web resource for test matrix collections.
In *Quality of Numerical Software*, pages 125–137. Springer, 1997.

[4] C. Croux and G. Haesbroeck.
Principal component analysis based on robust estimators of the covariance or correlation matrix: influence functions and efficiencies.
Biometrika, 87(3):603–618, 2000.

[5] T. A. Davis and Y. Hu.
The university of florida sparse matrix collection.
ACM Transactions on Mathematical Software (TOMS), 38(1):1, 2011.

[6] J. Demmel and A. McKenney.
A test matrix generation suite.
In *Courant Institute of Mathematical Sciences*. Citeseer, 1989.

[7] F. Dupros, F. De Martin, E. Foerster, D. Komatitsch, and J. Roman.
High-performance finite-element simulations of seismic wave propagation in three-dimensional nonlinear inelastic geological media.
Parallel Computing, 36(5):308–325, 2010.

References II

[8] M. Feit, J. Fleck, and A. Steiger.
Solution of the schrödinger equation by a spectral method.
Journal of Computational Physics, 47(3):412–433, 1982.

[9] A. Fender, N. Emad, S. Petiton, and M. Naumov.
Parallel modularity clustering.
Procedia Computer Science, 108:1793–1802, 2017.

[10] H. Galicher, F. Boillod-Cerneux, S. Petiton, and C. Calvin.
Generate very large sparse matrices starting from a given spectrum. in lecture notes in computer science, 8969, springer (2014).

[11] T. Sakurai, H. Tadano, T. Ikegami, and U. Nagashima.
A parallel eigensolver using contour integration for generalized eigenvalue problems in molecular simulation.
Taiwanese Journal of Mathematics, pages 855–867, 2010.

[12] X. Wu and S. G. Petiton.
A distributed and parallel asyn- chronous unite and conquer method to solve large scale non-hermitian linear systems.
In *In HPC Asia 2018: International Conference on High Performance Computing in Asia-Pacific Region, Tokyo, Japan*, Jan. 2018.

Thank you for your attentions!

Questions?