A Parallel Generator of Non-Hermitian Matrices computed from Known Given Spectra

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

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

[2]CRIStAL, Université de Lille, France

PMAA18, Zurich, Jun. 2018

Outline

Linear System Solvers and Spectra

When we solve the linear systems

$$Ax = b$$

by the Krylov Subspace methods, such as GMRES (Saad and Schultz (1986)), with A a non-Hermitian matrix. The spectra have more or less the impact during the procedure of resolution by these methods, such as:

1. Convergence Analysis;
2. Preconditioners;
3. Recyling of eigenvalues for a sequence of linear systems;
4. etc.

Requirement of large-scale matrix generator

Today:

- the linear problem size is increasing;
- the numerical methods should adjust to the coming exascale platforms.

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

- their spectra must be known and can be customized;
- they should be sparse, non-Hermitian and non-trivial;
- they could have a very high dimension to evaluate the algorithms on large-scale systems;
- they should be generated in parallel with low memory required during the procedure of generation.

Related works

The related work:

- Saad's SPARSKIT (Saad (1990));
- Tim Davis collection (Davis and Hu (2011));
- Matrix Market collection (Boisvert et al. (1997));
- Bai's collection (Bai et al. (1996))
- Galeri package of Trilinos to generate simple well-know finite element and finite difference matrices;
- J. Demmel's generation suite in 1989 to benchmark LAPACK (Demmel and McKenney (1989)), etc.

Only the method by Demmel generate matrices with given spectra, which can transfer the diagonal matrix into a dense matrix by the orthogonal matrices, and then reduce them to unsymmetric band ones by 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

Based on the preliminary theoretical of H. Galicher (Galicher et al. (2014)), 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, we select A to be a nilpotent matrix.

Nilpotent Matrix

The selected nilpotent matrix is given as:

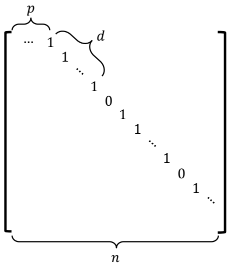


Figure: Nilpotent Matrix.

If $p = 1$, with $d \in \mathbb{N}^*$, or $p = 2$ with $d \in \mathbb{N}^*$ to be even, the nilpotency of A is $d + 1$.

SMG2S Algorithm

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 $M_o \in \mathbb{C}^{n\times n}$
2: Insert $Spec_{in}$ on the diagonal of $M_0$ and $M_0 = (2d-2)!M_0$
3: Generate the nilpotent matrix $A \in \mathbb{N}^{n\times n}$ with parameters $p$ and $d$
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: $M_t = \frac{1}{(2d-2)!}M_{2d-2}$
```

Matrix Generation Example

Through SMG2S, this nilpotent matrix can transfer an low band matrix to be a band matrix which have same spectrum.

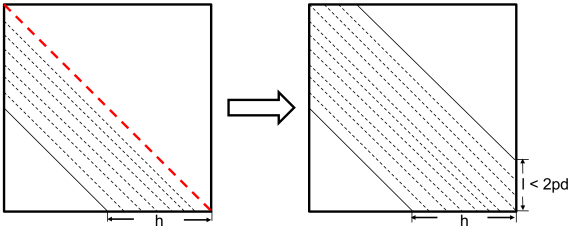


Figure: Matrix Generation Example.

Operation complexity is $max(\mathcal{O}(hdn), \mathcal{O}(d^2n))$. If $d \ll n$ and $h \ll n$, it turns out to be $\mathcal{O}(n)$ operations and memory space.

Parallel Implementation of CPUs and GPUs

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

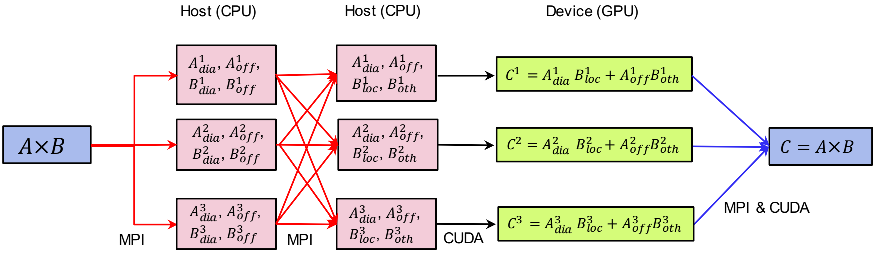

Host (CPU)
Host (CPU)
Device (GPU)
$A \times B$
$A^1_{dia}, A^1_{off}, B^1_{dia}, B^1_{off}$
$A^2_{dia}, A^2_{off}, B^2_{dia}, B^2_{off}$
$A^3_{dia}, A^3_{off}, B^3_{dia}, B^3_{off}$
$A^1_{dia}, A^1_{off}, B^1_{loc}, B^1_{oth}$
$A^2_{dia}, A^2_{off}, B^2_{loc}, B^2_{oth}$
$A^3_{dia}, A^3_{off}, B^3_{loc},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

Optimized Communication Implementation on CPUs

The implementation of SMG2S, especially the parallel SpGEMM kernel's communication can be specifically optimized based on the particular property of nilpotent matrix A.

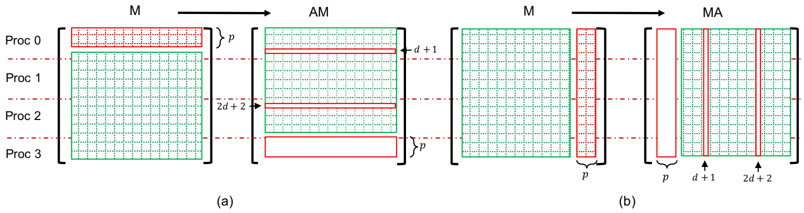


Figure: (a) AM operation; (b) MA operation.

Outline

Experimental hardware environment

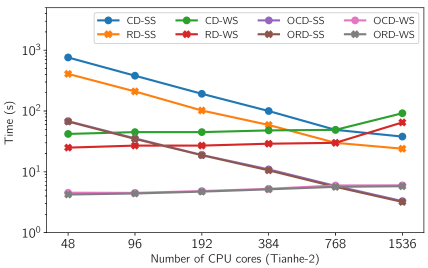
We implement SMG2S on the supercomputers *Tianhe-2* and *Romeo*. The node specfication for the two platforms is given as following:

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

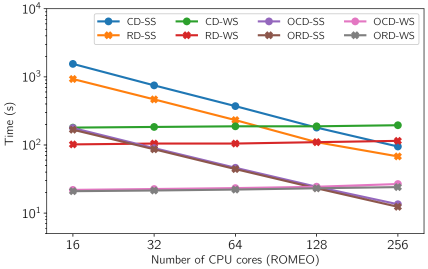
Machine Name	ROMEO	Tianhe-2
Nodes Number	BullX R421 $\times$ 130	16000 $\times$ nodes
Mother Board	SuperMicro X9DRG-QF	Specific Infiniband
CPU	2$\times$Intel Ivy Bridge 8 cores 2.6 GHz	2$\times$Intel Ivy Bridge 12 cores 2.2 GHz
Memory	DDR3 32GB	DDR3 64GB
Accelerator	NVIDIA GPU Tesla K20X $\times$ 2	Intel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

Scalability and Speedup Evaluation I

The scaling and speedup evaluations are given as:

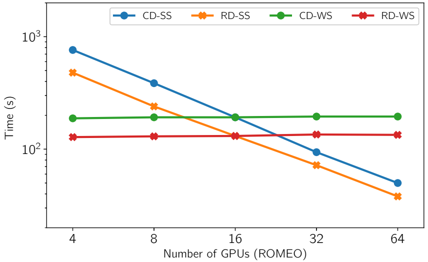


(a) Strong and Weak Scaling of SMG2S on Tianhe-2

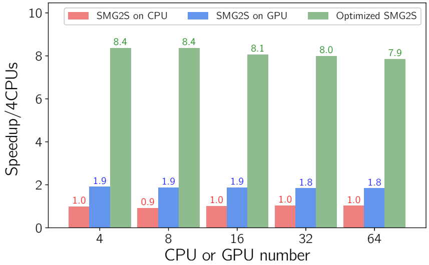


(b) Strong and Weak Scaling of SMG2S on ROMEO

Scalability and Speedup Evaluation II



(c) Strong and Weak Scaling of SMG2S on ROMEO with multiple GPUs



(d) Speedup of different implementation

Outline

Verification method

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)

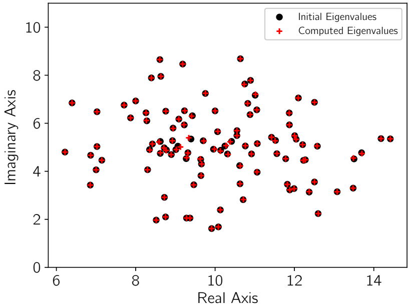
```
1: $y = v_0$
2: for $i = 1, 2, 3 \cdots$ do
3:     $\theta = ||y||_\infty$, $v = y/\theta$
4:     Solve $(A - \sigma I)y = v$
5: end for
```

Check error

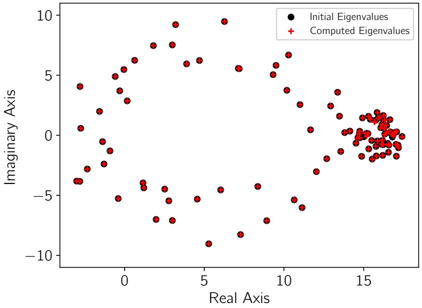
$$error = \frac{||Av' - \lambda v'||}{||Av'||}$$

Verification results I

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

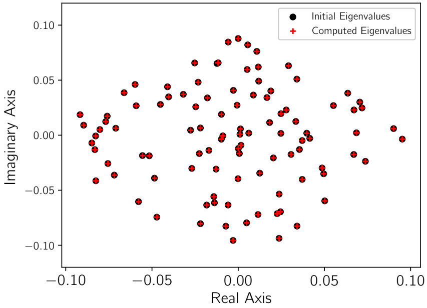


(e) Spec1: Clustered Eigenvalues I

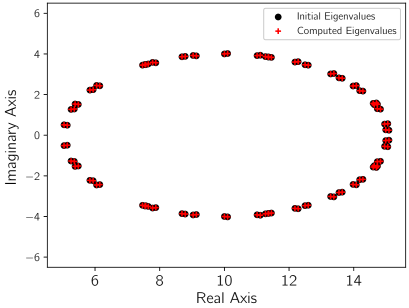


(f) Spec2: Dominant Clustered Eigenvalue II

Verification results II



(g) Spec3: Clustered Eigenvalues III



(h) Spec4: Conjugate and Closest Eigenvalues

Figure: Verification using Different Types of Spectra.

Outline

Workflow

An interface can be provided to PETSc, Trilinos, and any public or personal parallel solvers, which can restore the distributed data into the data structures of different libraries. This feature can significantly reduce the I/O of applications and improve their efficiency to evaluate the numerical methods.

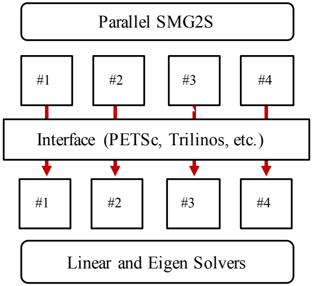


Figure: SMG2S Workflow and Interface.

An Example

We have also using SMG2S to evaluate different iterative methods for solving non-Hermitian linear systems.

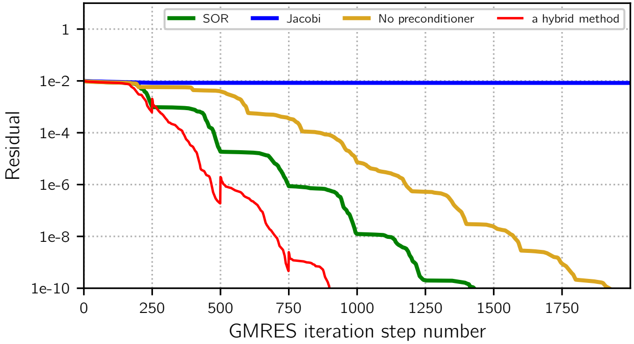


Figure: Convergence Comparision Example.

Outline

Conclusion and Perspectives

Then

1. SMG2S is a method to generate large-scale non-Hermitian matrices with good scalability and capacility to keep accuracy of given spectra;
2. Interfaces to more scientific libraries should be implemented;
3. Beta version is available on Github: https://github.com/brunowu/SMG2S.git;
4. The package of the software will be finished soon;
5. It will be shared with researchers around world from related background.

The related talk:
X. Wu and S.G. Petiton: A Parallel Generator of Non-Hermintian Matrices Computed from Known Given Spectra. SIAM PP18 in Tokyo, Japan.
The related paper:
X. Wu and S.G. Petiton: A Parallel Generator of Non-Hermitian Matrices computed from Given Spectra. (Submitted to IEEE CLUSTER2018).

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 funded by the the German-Japanese-French project MYX project of French National Research Agency (ANR) under the SPPEXA framework.

Thank you for your attentions!

Questions?