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

Xinzhe WU[1,2] Serge G. Petiton[1,2] Yutong Lu[3]

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

[2]CRIStAL, Université de Lille, France

[3]National Supercomputing Center in Guangzhou, Sun Yat-sen University, China

VECPAR18

São Pedro, Brazil, 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. Deflation of eigenvalues;
4. Recyling of eigenvalues for a sequence of linear systems;
5. 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 good scalability performance and low memory requirement 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 work 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:

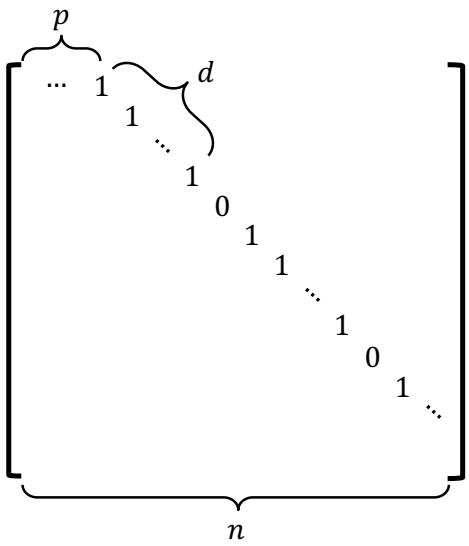


Figure: Nilpotent Matrix.

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: $Spec_{in} \in \mathbb{C}^n$, p, 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 $M_0 = (2d)!M_0$
3: Generate the nilpotent matrix $A \in \mathbb{N}^{n \times n}$ with parameters p and d
4: **for** $i = 0, \cdots, 2d-1$ **do**
5: $\quad M_{i+1} = M_i + (\prod_{k=i+1}^{2d} k)(\widetilde{A_A})^i(M_0)$
6: **end for**
7: $M_t = \frac{1}{(2d)!} M_{2d}$

Matrix Generation Example

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

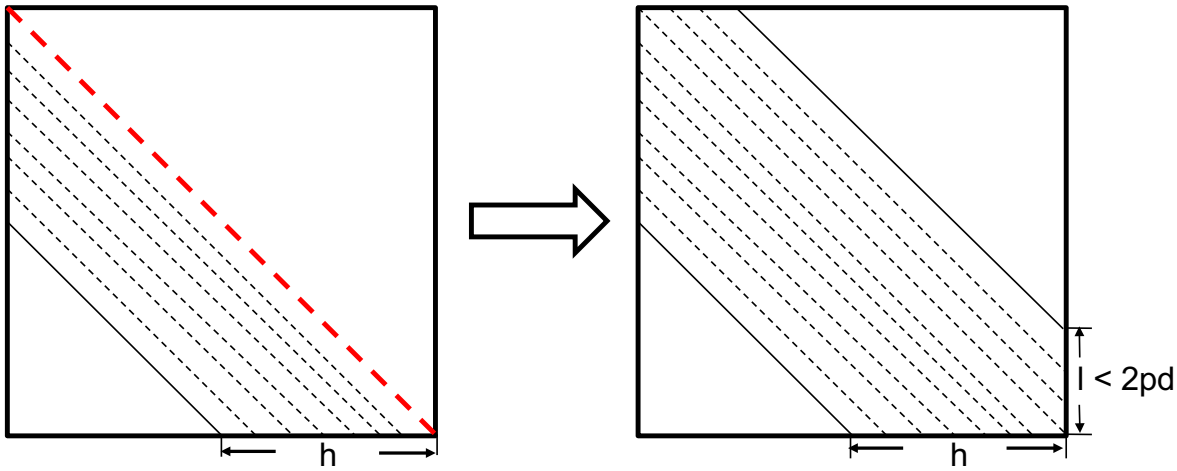


Figure: Matrix Generation Example.

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

Matrix Generation Example

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

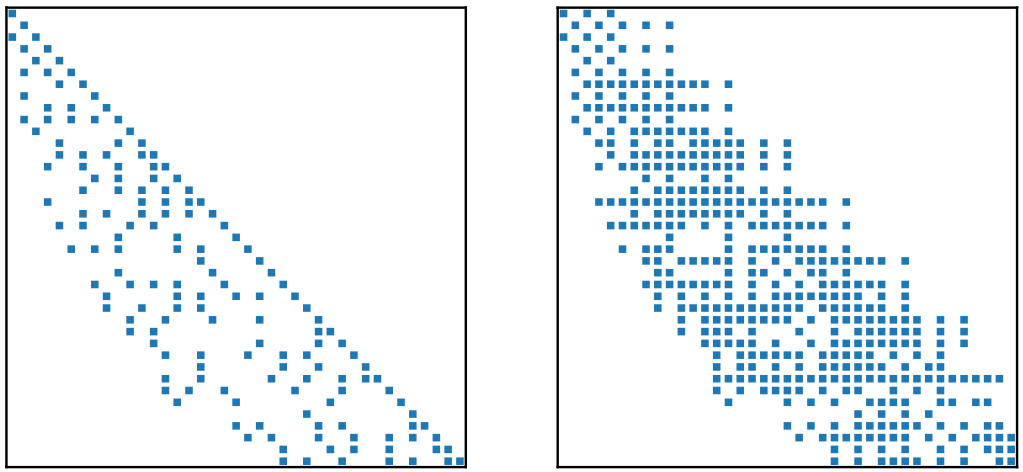


Figure: Matrix Generation Sparsity Pattern.

Operation complexity is $max(\mathcal{O}(hdn), \mathcal{O}(d^2 n))$. 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.

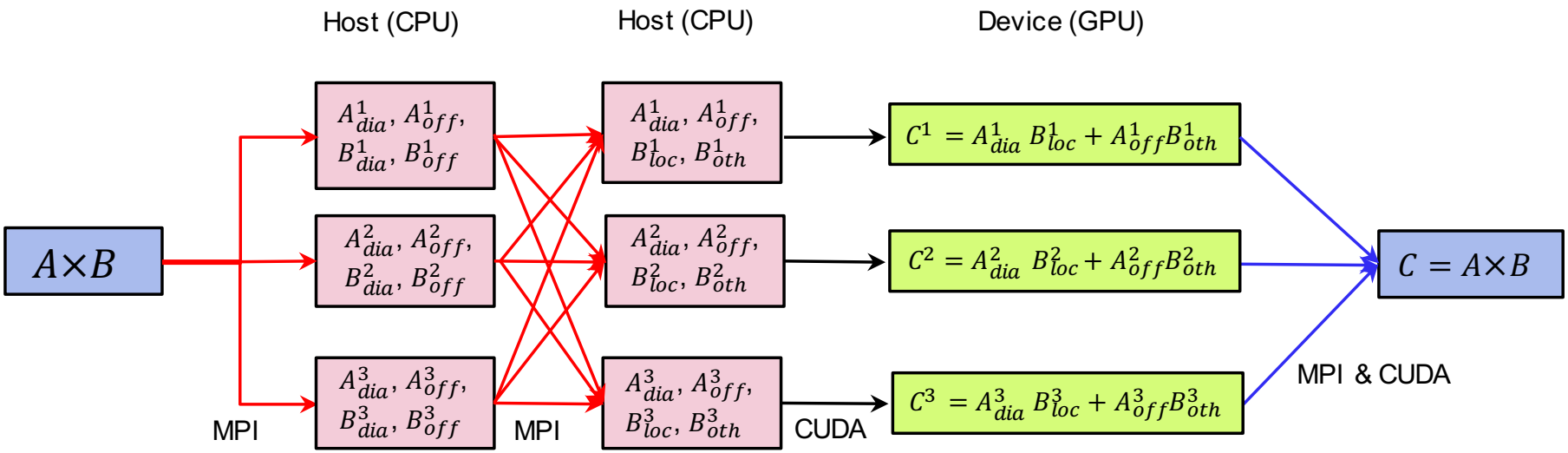


Figure: The structure of a CPU-GPU implementation of SpGEMM, where each GPU is attached to a CPU. The GPU is in charge of the computation, while the CPU handles the MPI communication among processes.

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.

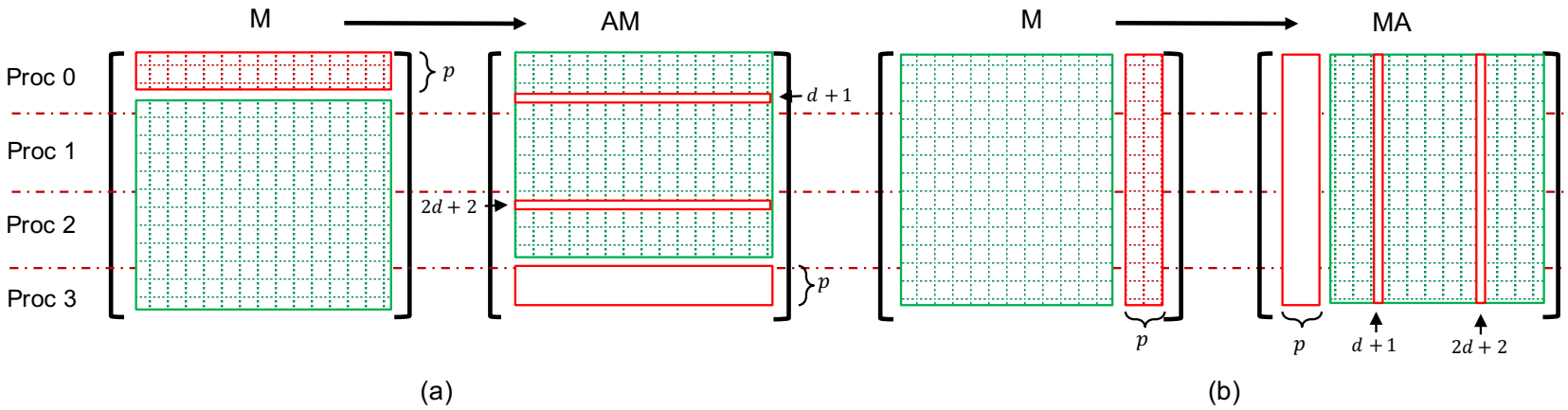


Figure: (a) AM operation; (b) MA operation.

Outline

Experimental hardware environment

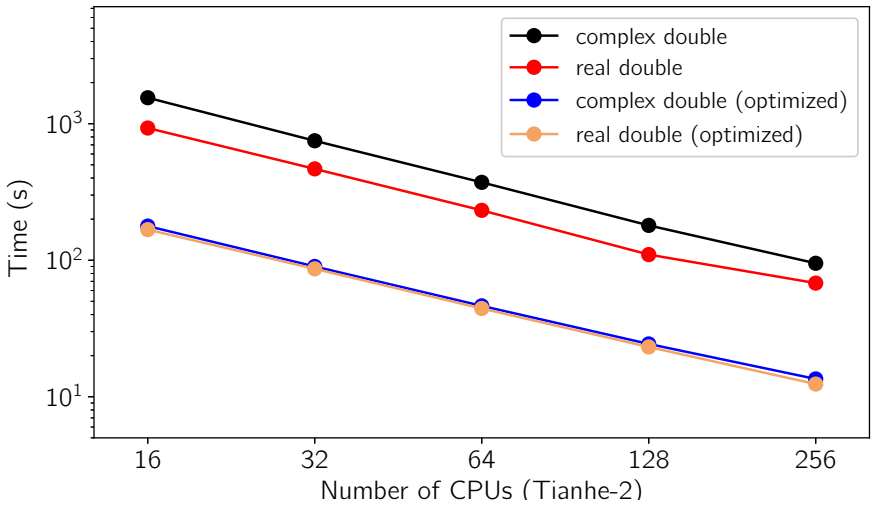
We implement SMG2S on the supercomputers *Tianhe-2* and *Romeo*. The node specfication for the two platforms is given as following:

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

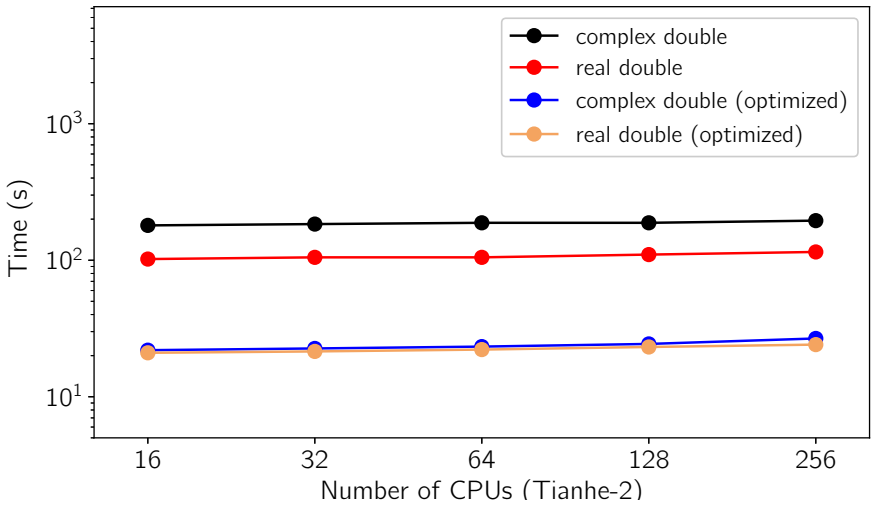
Machine Name	ROMEO	Tianhe-2
Nodes Number	BullX R421 × 130	16000 × nodes
Mother Board	SuperMicro X9DRG-QF	Specific Infiniband
CPU	2×Intel Ivy Bridge 8 cores 2.6 GHz	2×Intel Ivy Bridge 12 cores 2.2 GHz
Memory	DDR3 32GB	DDR3 64GB
Accelerator	NVIDIA GPU Tesla K20X × 2	Intel Knights Corner × 3

Scalability and Speedup Evaluation I

The scaling evaluations of CPUs on ROMEO are given as:



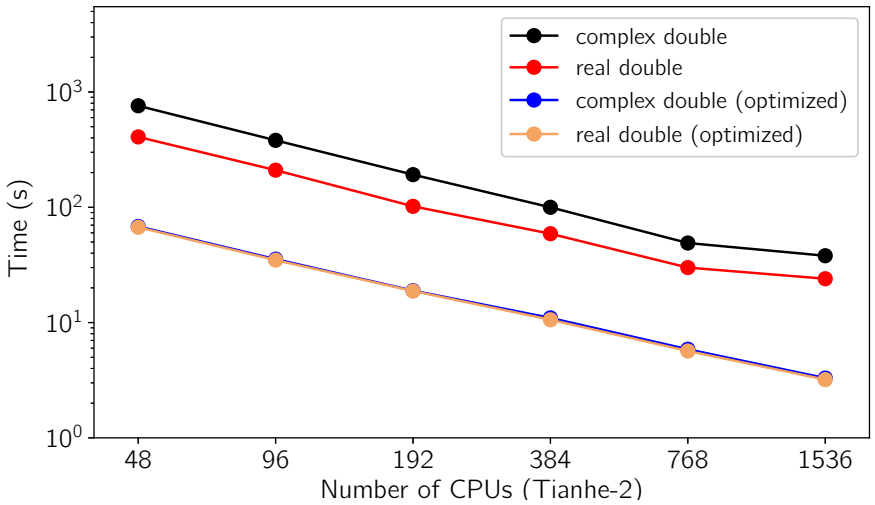
(a) CPU strong scaling on ROMEO.



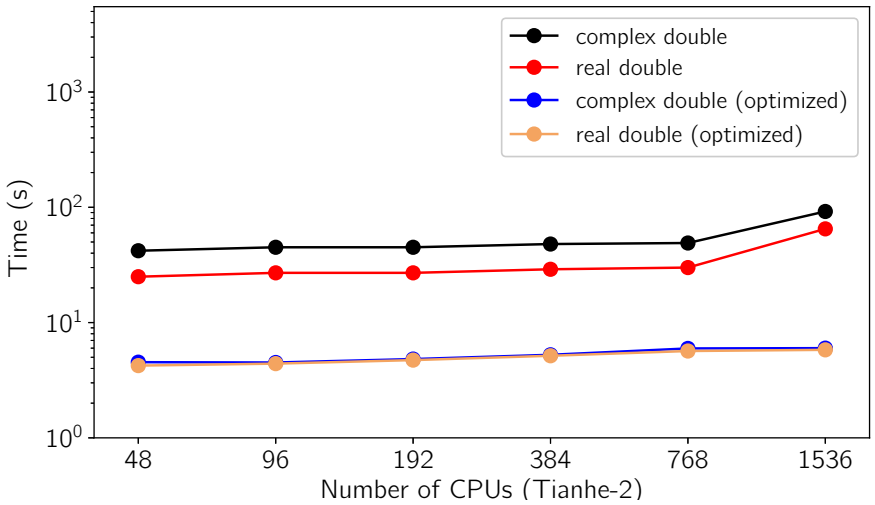
(b) CPU weak scaling on ROMEO.

Scalability and Speedup Evaluation II

The scaling evaluations of CPUs on Tianhe-2 are given as:



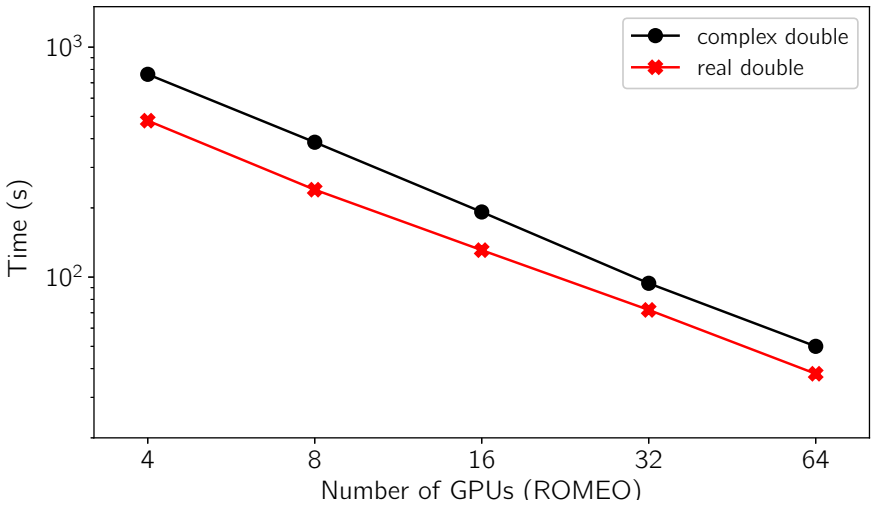
(c) CPU strong scaling on Tianhe-2.



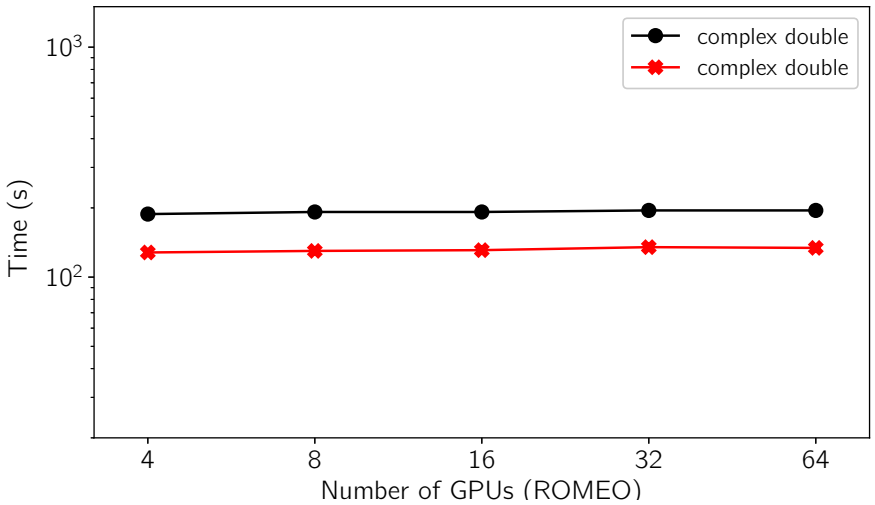
(d) CPU weak scaling on Tianhe-2.

Scalability and Speedup Evaluation III

The scaling evaluations of GPUs on ROMEO are given as:



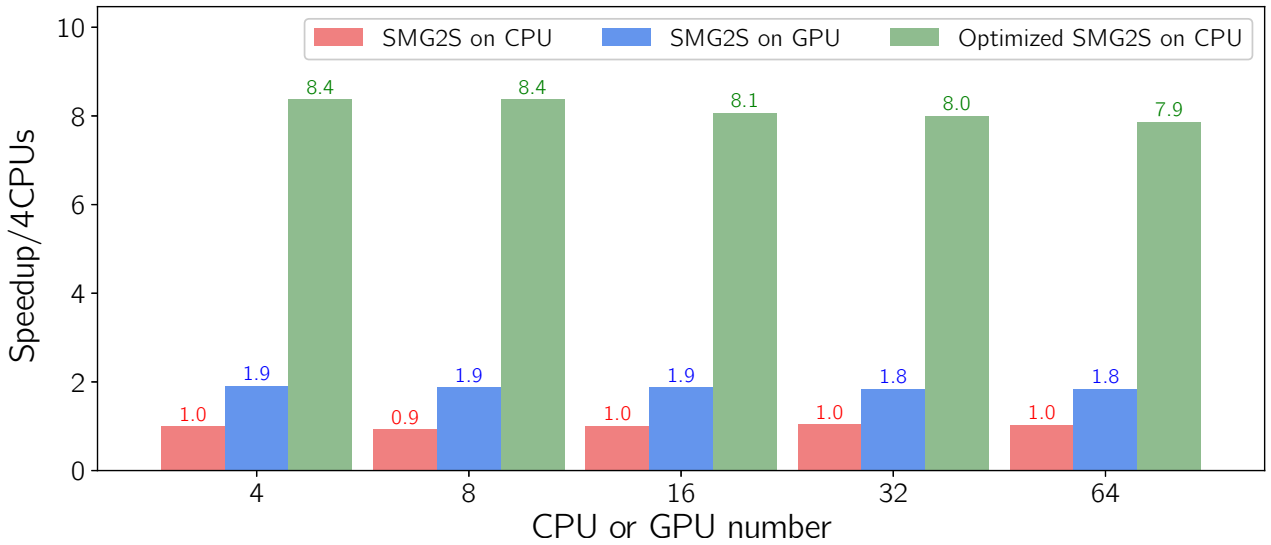
(e) GPU strong scaling on ROMEO.



(f) GPU weak scaling on ROMEO.

Scalability and Speedup Evaluation IV

The speedup evaluations of different implementations are given as:



(g) 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)

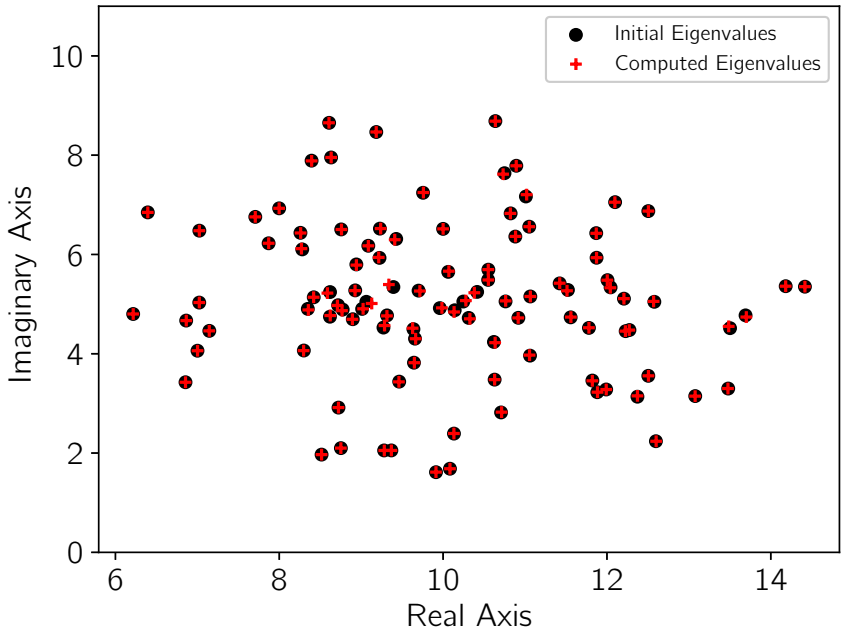
1: $y = v_0$
2: **for** $i = 1, 2, 3 \cdots$ **do**
3: $\quad \theta = ||y||_\infty$, $v = y/\theta$
4: $\quad$ Solve $(A - \sigma I)y = v$
5: **end for**

Check error

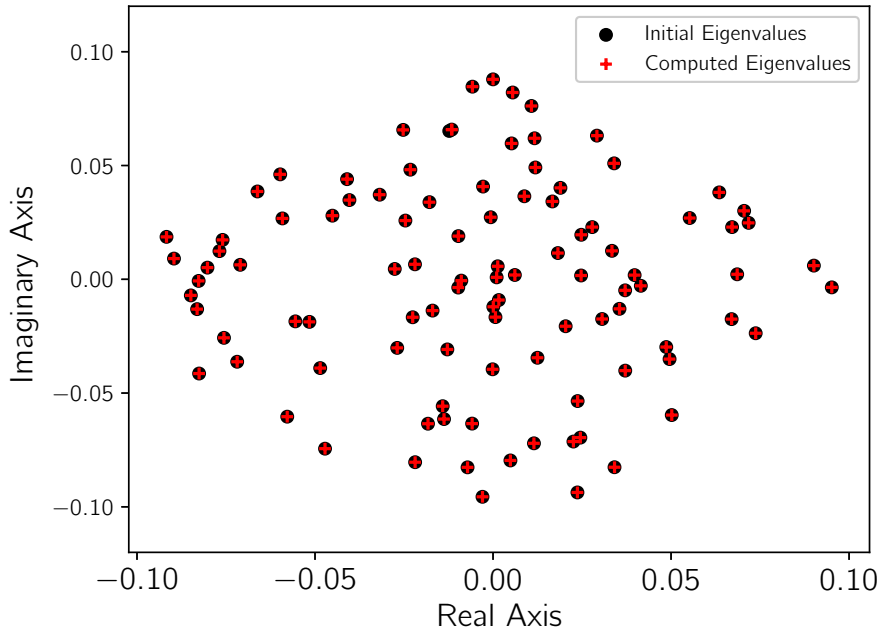
$$error = \frac{||Av' - \lambda v'||}{||Av'||}$$

Verification results I

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



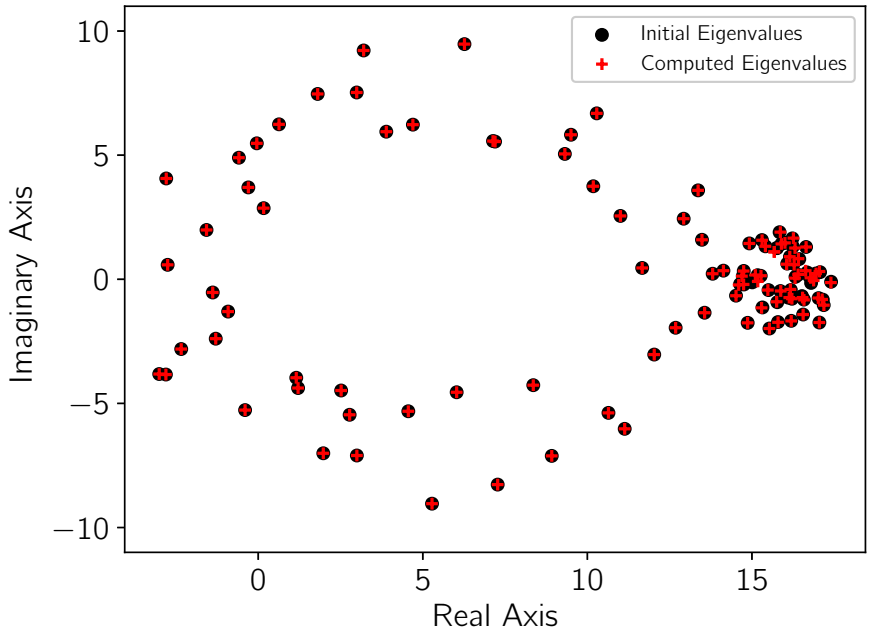
(h) Clustered Eigenvalues I: acceptance = 93%, max *error* $= 2 \times 10^{-2}$



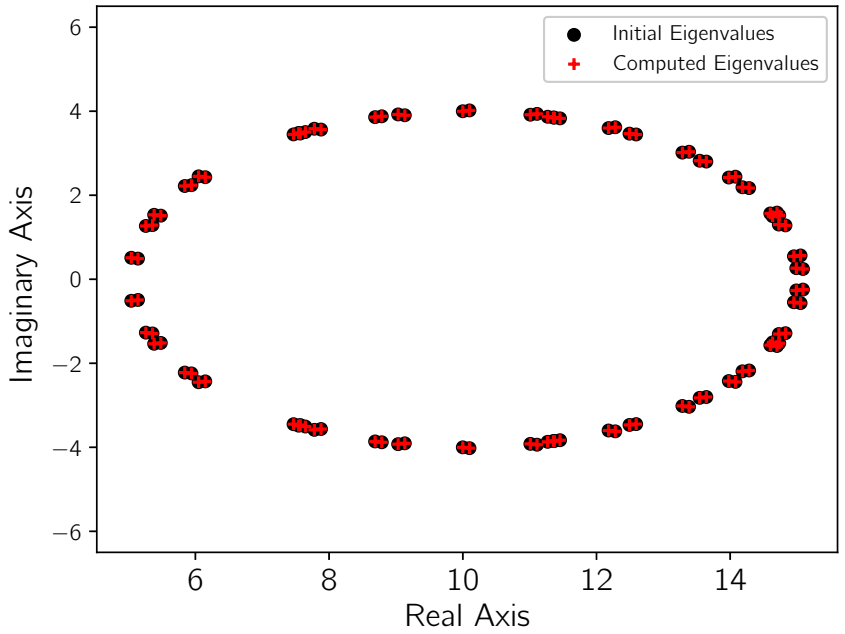
(i) Clustered Eigenvalues II: acceptance = 100%, max *error* $= 7 \times 10^{-5}$

Verification results II

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



(j) Dominant Clustered Eigenvalues: acceptance = 94%, max *error* $= 3 \times 10^{-2}$



(k) Conjugate and Closest Eigenvalues: acceptance = 100%, max *error* $= 3 \times 10^{-7}$

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.

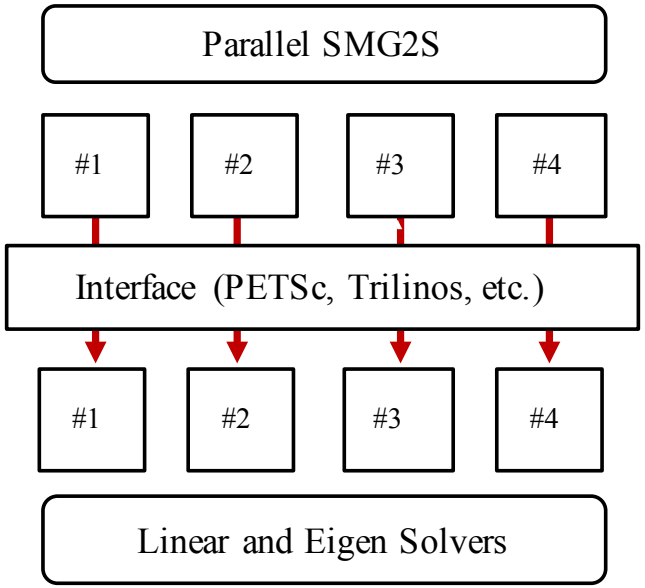


Figure: SMG2S Workflow and Interface.

An Example

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

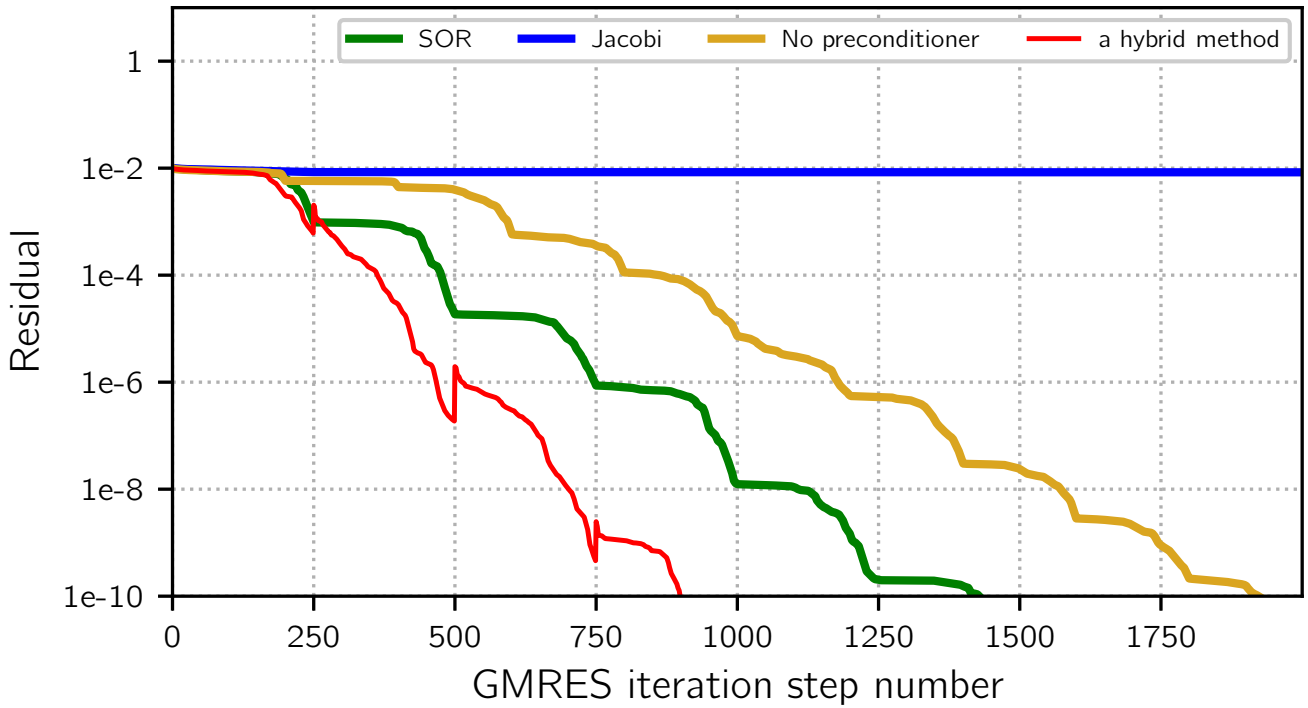


Figure: Convergence Comparison using a matrix generated by SMG2S.

Outline

How to Get SMG2S

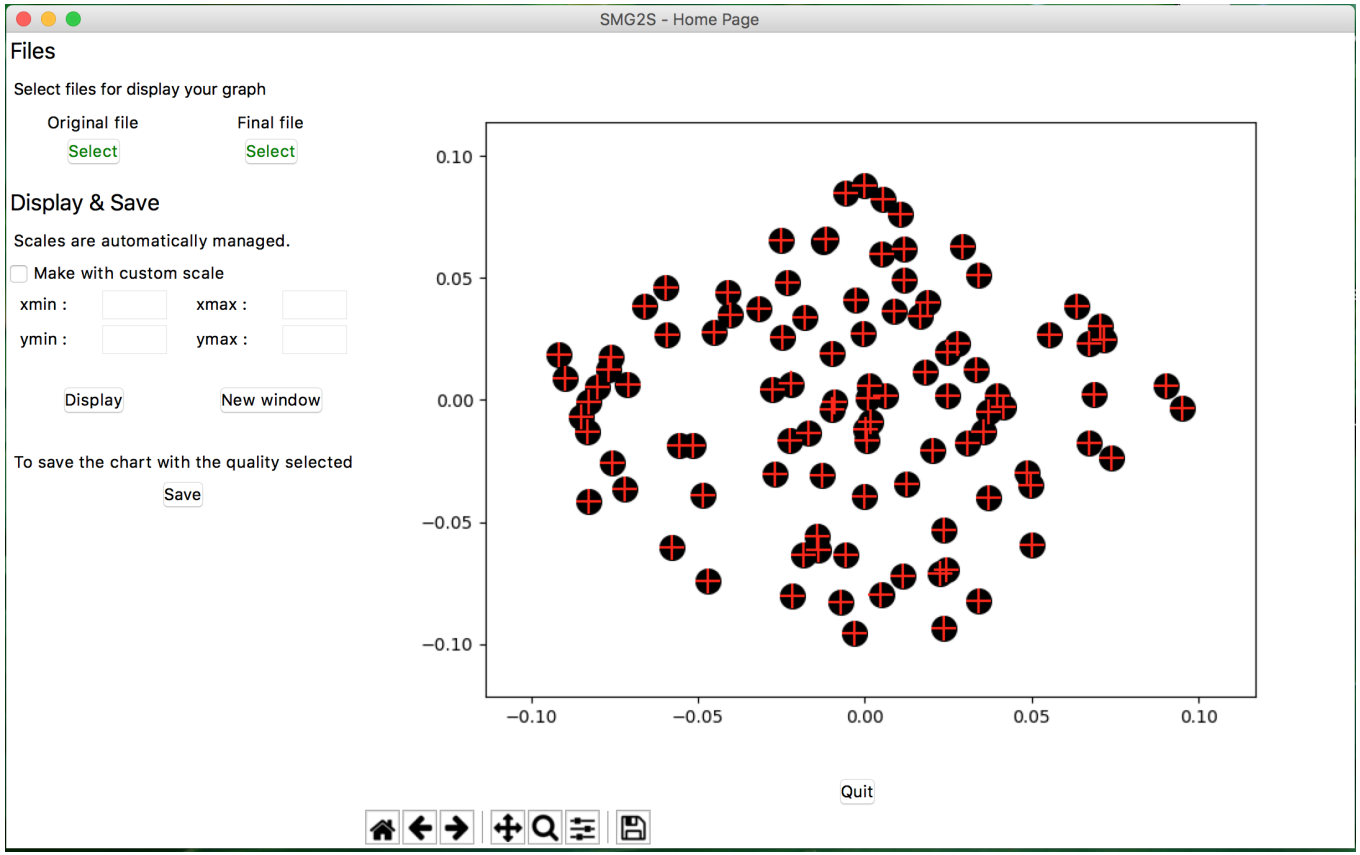
1. SMG2S is an open source software released under the GNU Lesser General Public License v3.0.
2. Website: https://smg2s.github.io.
3. Development version is available on Github: https://github.com/SMG2S.
4. Documentation: https://smg2s.github.io/files/smg2s-manual.pdf.
5. Until now, SMG2S provides the interface to scientific computational softwares PETSc and Trilinos/Tpetra and also C and Python programming languages.

Contact us:

xinzhe.wu@ed.univ-lille1.fr and serge.petiton@univ-lille1.fr

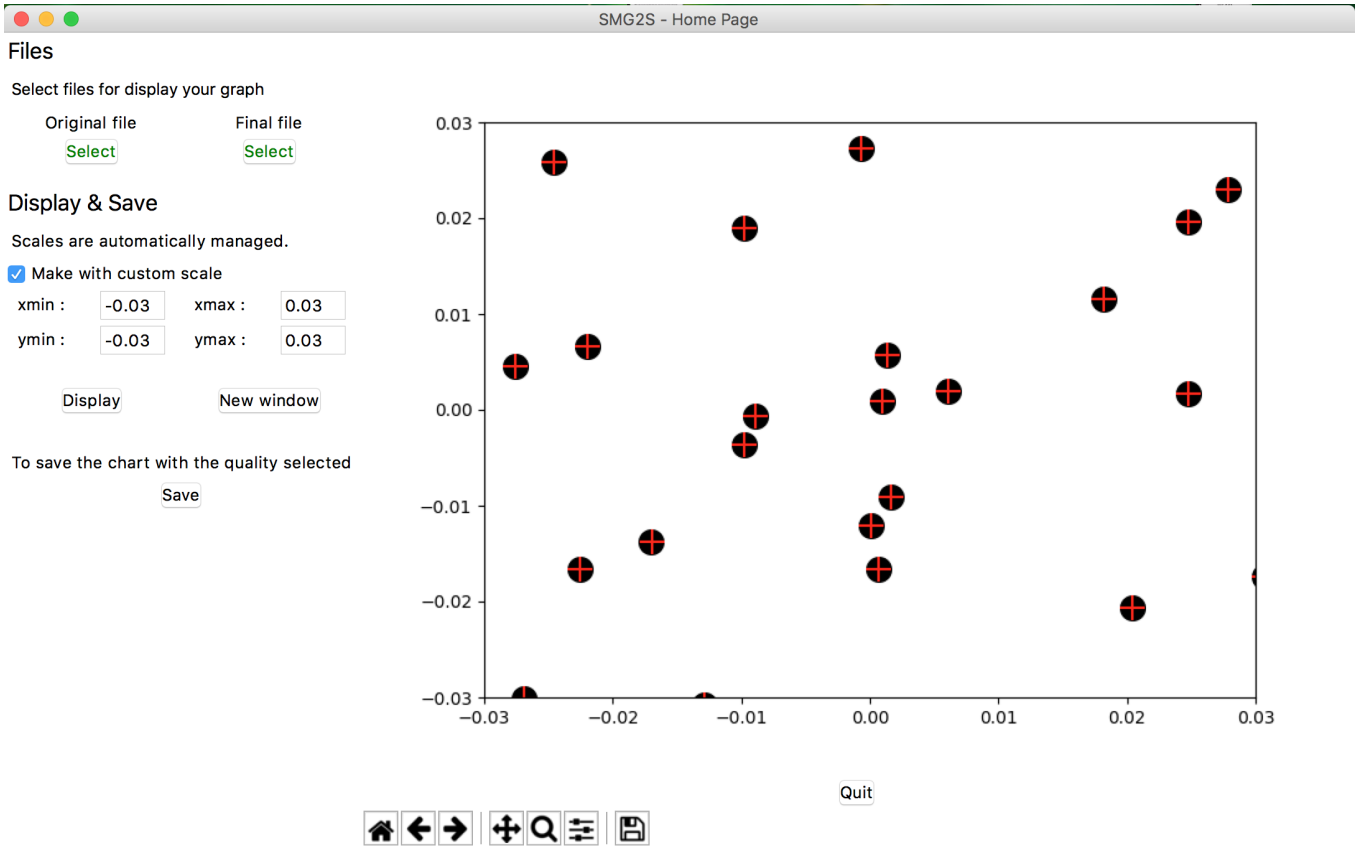
Graphic User Interface

SMG2S provides the Graphic User Interface for the eigenvalues accuracy verification.



Graphic User Interface

SMG2S provides the Graphic User Interface for the eigenvalues accuracy verification.



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. Interface to Fotran can be supported in future.

The related talks:
X. Wu and S.G. Petiton: A Parallel Generator of Non-Hermintian Matrices Computed from Known Given Spectra. SIAM PP18 in Tokyo, Japan.
X. Wu and S.G. Petiton: A Parallel Generator of Non-Hermitian Matrices Computed from Given Spectra. PMAA18 in Zürich, Switzerland.

Acknowledgement

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

Thank you for your attentions!

Questions?

Theorem

Theorem

Let's consider the matrices $A \in \mathbb{C}^{n\times n}$, $M_0 \in \mathbb{C}^{n\times n}$, $n \in \mathbb{N}^*$. *If* M *verifies:*

$$\begin{cases} \dfrac{dM(t)}{dt} = AM(t) - M(t)A, \\ M(t=0) = M_0. \end{cases}$$

Then the matrices $M(t)$ *and* M_0 *are similar,* $\forall A \in \mathbb{C}^{n\times n}$.

Proof.

Denote respectively $\sigma(M_0)$ and $\sigma(M_t)$ the spectra of M_0 and M_t. If M_0 is a diagonalisable matrix, $\forall \lambda \in \sigma(M_0)$, it exists an eigenvector $v \neq 0$ satisfies the relation:

$$M_0 v = \lambda v. \tag{4}$$

Denote $v(t)$ by the matrix $B \in I_n$:

$$v(t) = B_t v = e^{tA} B v. \tag{5}$$

We can get:

$$\begin{aligned} \frac{d(M_t v(t) - \lambda v(t))}{dt} &= \frac{dM_t}{dt} v(t) + M_t \frac{dv(t)}{dt} - \lambda \frac{dv(t)}{dt} \\ &= A(M_t v(t) - \lambda v(t)) + \lambda A v(t) \\ &\quad - M_t A v(t) + M_t \frac{dB_t}{dt} v - \lambda \frac{dB_t}{dt} v. \end{aligned} \tag{6}$$

Proof.

With the definition of B_t in Equation (5), we have:

$$\frac{dB_t}{dt} = AB_t. \tag{4}$$

Thus the Equation (6) can be simplified as

$$\frac{d(M_t v(t) - \lambda v(t))}{dt} = A(M_t v(t) - \lambda v(t)). \tag{5}$$

The initial condition for the above Equation is:

$$\begin{aligned} M_t v(t) - \lambda v(t)|_{t=0} &= M_0 Bv - \lambda Bv \\ &= M_0 v - \lambda v = 0. \end{aligned} \tag{6}$$

Hence the solution of this differential Equation is 0 and $\forall \lambda \in \sigma(M_0)$, we have $\lambda \in \sigma(M_t)$. Since $dim(M_0) = dim(M_t)$, we have $\sigma(M_0) = \sigma(M_t)$. M_0 and M_t are similiar with same eigenvalues, but different eigenvectors.

□