SMG2S User Manual

Xinzhe Wu

August 23, 2018

Contents

Chapter 1

Introduction

1.1 Getting Started

SMG2S (Scalable Matrix Generator with Given Spectrum) is a software which provides to generator the non-Hermitian Matrices with User-customized eigenvalues. SMG2S is implemented in parallel based on MPI (Message Passing Interface) and C++11 to support efficiently the generation of test matrices on distributed memory platforms.

Iterative linear algebra methods are important for the applications in various fields. The analysis of the iterative method behaviors is complex, and it is necessary to evaluate their convergence to solve extremely large non-Hermitian eigenvalue and linear problems on parallel and/or distributed machines. This convergence depends on the properties of spectra. Thus, we propse SMG2S to generate large matrices with known spectra to benchmark these methods. The generated matrices are non-Hermitian and non-trivial, with very high dimension. SMG2S can generate the non-Hermitian matrices with user-custormized eigenvalues.

The ability of SMG2S to keep the accuracy of given spectrum can be verified by the functionality proposed inside SMG2S. This function is based the shift inverse power method. SMG2S gives also an graphic user interface to compare the given and final spectral distribution for the verification.

We will describe the following subset of the SMG2S.

- **Parallel Vector and Matrix:**
- **Nilpotency Matrix Object:**
- **Generating Matrix with predescribed eigenvalues:**
- **Interface to Other Languages/Libraries:**
- **Verification of Eigenvalues of Generated Matrix:**

1.2 Installation

To obtain SMG2S, please follow the instructions at the SMG2S download page: https://github.com/brunowu/SMG2S.

Prerequisites:

- C++ Compiler with **c++11** support;
- Cmake (version minimum 3.6);
- (Optional) PETSc and SLPEc are necessary for the verification of the ability to keep the given spectrum.

Int the main directory:

```
cmake . –DCMAKE_INSTALL_PREFIX=${INSTALL_DIRECTORY}
```

The main.cpp will generate an exectutable smg2s.exe to demonstrate a minimum sample :

```
make
```

The main part of SMG2S is a colllection of header files. Install the header files into ${INSTALL_DIRECTORY}

```
make install
```

For testing:

```
make test
```

The output of test should be like:

```
Running tests...
Test project /User/name/SMG2S
Start 1: Test_Size_10000_w_proc1
1/4 Test #1: Test_Size_10000_w_proc1 .. Passed 1.20 sec
Start 2: Test_Size_20000_w_proc2
2/4 Test #2: Test_Size_20000_w_proc2 .. Passed 1.22 sec
Start 3: Test_Size_10000_s_proc1
3/4 Test #3: Test_Size_10000_s_proc1 .. Passed 1.20 sec
Start 4: Test_Size_10000_s_proc2
4/4 Test #4: Test_Size_10000_s_proc2 .. Passed 0.66 sec

100% tests passed, 0 tests failed out of 4

Total Test time (real) = 4.29 sec
```

1.3 Copyright and Licensing of SMG2S

SMG2S is a open source software published under the GNU Lesser General Public License v3.0. SMG2S can be redistributed and modified under the terms of this license.

SMG2S is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. SMG2S is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with SMG2S. If not, see ¡http://www.gnu.org/licenses/¿.

1.4 Programming Language in SMG2S

SMG2S is a collection of templated header files written in C++. The wrappers to C and Python codes are provided. The users of PETSc or Trilinos can directly use SMG2S with the interfaces implemented inside.

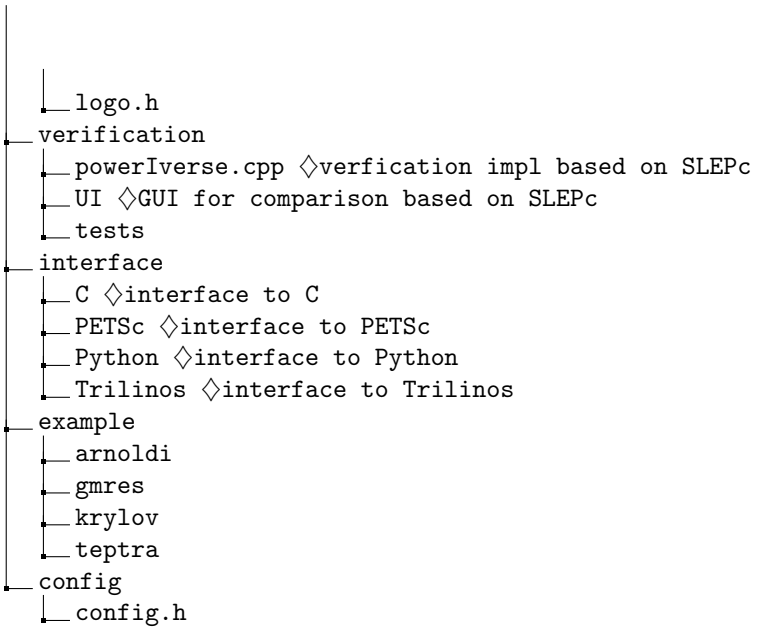
1.5 Referencing SMG2S

@article{galichergenerate, title={Generate Very Large Sparse Matrices Starting from a Given Spectrum}, author={Galicher, Hervé and Boillod-Cerneux, France and Petiton, Serge and Calvin, Christophe} }

1.6 Directory Structure

SMG2S is a collection of header files, and its directory structure is given as follows:

```
SMG2S
├── parVector
│   ├── parVectorMap.h ♢implementation of distributed vector map
│   └── parVector.h ♢implementation of distributed vector
├── parMatrix
│   ├── MatrixCSR.h ♢implementation of serial CSR Matrix
│   └── parMatrixSparse.h ♢distributed sparse matrix
├── smg2s
│   ├── specGen.h ♢Function to provide given spectrum
│   └── smg2s.h ♢ implementation of smg2s generator
└── utlis
    ├── MPI_DataType.h
    └── utlis.h
```



1.7 List of SMG2S Contributors

This is the list of SMG2S contributors:

Xinzhe Wu	main constributor	xinzhe.wu@cea.fr
Serge Petiton	Supervisor	serge.petiton@univ-lille1.fr
Quentin Petit	GUI Implementation (Intern)	quentin.petit@polyetch-lille.net

Chapter 2

Templated SMG2S Parallel Matrix and Vector

2.1 Parallel Vector

2.1.1 Vector Map

2.1.2 Creating a Distributed Vector

2.1.3 Parallel Matrix

2.1.4 Matrix Map

2.1.5 Creating a Distributed Matrix

Chapter 3

Templated Nilpotency Matrix Object

Chapter 4

Generating Matrix with SMG2S

4.1 Generation Workflow

4.2 Example

Include the head file:

```
#include <smg2s/smg2s.h>
```

Generate the Nilpotent Matrix Object:

```
Nilpotency<int> nilp;
nilp.NilpType1(length, probSize);
```

Create the parallel Sparse Matrix Object Mt:

```
parMatrixSparse<std::complex<double>,int> *Mt;
```

Generate a new matrix by SMG2S:

```
MPI_Comm comm; //working MPI Communicator
Mt = smg2s<std::complex<double>,int>(probSize, nilp,
lbandwidth, spectrum, comm);
```

Here, the **probsize** parameter represent the matrix size, **nilp** is the nilpotency matrix object that we have declared previously, **lbandwidth** is the bandwidth of lower-diagonal band. **spectrum** is the file path of spectra file, if **spectrum** is set as " ", SMG2S will use the mechanism inside to generate the spectral distribution. **comm** is the basic object used by MPI to determine which processes are involved in a communication.

The given spectra file is in **pseudo-Matrix Market Vector format**. For the complex eigenvalues, the given spectrum is stored in three columns, the first

column is the coordinates, the second column is the real part of complex values, and the third column is the imaginary part of complex values.

```
%%MatrixMarket matrix coordinate complex general
3 3 3
1 10 6.5154
2 10.6288 3.4790
3 10.7621 5.0540
```

For the eigenvalues values, the given spectrum is stored in two columns, the first column is the coordinates, the second column is related values.

```
%%MatrixMarket matrix coordinate complex general
3 3
1 10
2 10.6288
3 10.7621
```

Chapter 5

Interface to Other Languages/Libraries

5.1 Interface to C

SMG2S install command will generate a shared library libsmg2s.so (libsmg2s2c.dylib on OS X platform) into ${INSTALL_DIRECTORY}/lib. It can be used to profit the C wrapper of SMG2S.

The way to use:

Include the header file:

```
#include <interface/C/c_wrapper.h>
```

create Nilpotency object :

```
struct NilpotencyInt *n;
n = newNilpotencyInt();
NilpType1(n, 2, 10);
```

After that, you need to create the parallel Sparse Matrix Object Mt like this :

```
struct parMatrixSparseDoubleInt *m;
m = newParMatrixSparseDoubleInt();
```

Generate by SMG2S :

```
smg2s(m, 10, n, 3 ," " ,MPI_COMM_WORLD);
```

Release Nilpotency Object and parMatrixSparse Object :

```
smg2s(m, 10, n, 3 ," " ,MPI_COMM_WORLD);
```

5.2 Interface to Python

SMG2S uses SWIG to generate the wrapper of SMG2S to Python. Generate the shared library and install the python module of smg2s.

```
cd ./interface/python;
mpicxx -fpic -c smg2s_wrap.cxx  -I/apps/python/3/    \
include/python3.5m -std=c++0x
mpicxx -shared smg2s_wrap.o -o _smg2s.so
python setup.py install
```

Before the utilisation, make sure that **mpi4py** is installed.
This is a little example of usge :

```
from mpi4py import MPI
import smg2s
import sys

size = MPI.COMM_WORLD.Get_size()
rank = MPI.COMM_WORLD.Get_rank()
name = MPI.Get_processor_name()

sys.stdout.write(
"Hello, World! I am process %d of %d on %s.\n"
% (rank, size, name))

if rank == 0:
    print ('INFO ]> Starting ...')
    print("INFO ]> The MPI World Size is %d" %size)

#bandwidth for the lower band of initial matrix
lbandwidth = 3

#create the nilpotent matrix
nilp = smg2s.NilpotencyInt()

#setup the nilpotent matrix:
nilp.NilpType1(2,10)

Mt = smg2s.parMatrixSparseDoubleInt()

#Generate Mt by SMG2S
#vector.txt is the file that stores the given    \
spectral distribution in local filesystem.
Mt=smg2s.smg2sDoubleInt(10,nilp,lbandwidth,      \
"vector.txt", MPI.COMM_WORLD)
```

5.3 Interface to PETSc

SMG2S provides the interface to scientific computational softwares PETSc/SLEPc.

The way of Usage:

Include header file: Include the header file:

```
#include <interface/PETSc/petsc_interface.h>
```

Create parMatrixSparse type matrix :

```
parMatrixSparse<std::complex<double>,int> *Mt;
```

Restore this matrix into CSR format :

```
Mt->Loc_ConvertToCSR();
```

Create PETSc MAT type :

```
MatCreate(PETSC_COMM_WORLD,&A);
```

Convert to PETSc MAT format :

Create PETSc MAT type :

```
A = ConvertToPETSCMat(Mt);
```

5.4 Interface to Trilinos/Teptra

SMG2S is able to convert its distributed to the CSR one-dimensional distributed matrix defined by Teptra in Trilinos.

The way of usage:

Include header file

```
#include <interface/Trilinos/trilinos_interface.hpp>
```

Create parMatrixSparse type matrix :

```
parMatrixSparse<std::complex<double>,int> *Mt;
```

Create Trilinos/Teptra MAT type :

```
parMatrixSparse<std::complex<double>,int> *Mt;
```

Convert to Trilinos MAT format :

```
K = ConvertToTrilinosMat(Mt);
```

Here is a full example of Trilinos.

Chapter 6

Verification of Eigenvalues

6.1 Prerequisites

6.2 Verifcation by Shifted Inverse Power Method

run the verification script:

```
#!/bin/bash
EXEC=./powerInverse.exe
N=100
L=10
TEST_TOL=0.00001
DEGREE=4

LENGTH=$(awk 'NR==2{print $1}' vector.txt)
echo "Test Eigenvalues number = "${LENGTH}

for((i=3;i<=${LENGTH}+2;i++))
do
real=$(awk 'NR=='$i'{print $2}' vector.txt)
imag=$(awk 'NR=='$i'{print $3}' vector.txt)
srun -n 1 ${EXEC} -n ${N} -l ${L} -eps_monitor_conv    \
-eps_power_shift_type constant -st_type sinvert      \
-exact_value ${real}+${imag}i -test_tol ${TEST_TOL}    \
-degree ${DEGREE}
done
```

6.3 Script for result cleaning

```
#!/bin/bash

grep "@> The eigenvalue" $1 > tmp.txt
awk '{print $5 " " $7 }' tmp.txt > tmp2.txt
awk '{print substr($0, 1, length($0)-1)}' tmp2.txt \
> tmp3.txt

awk '{print NR  " " $0}' tmp3.txt > tmp4.txt
NB=`wc -l tmp4.txt | awk '{print $1}'`
awk 'BEGIN{print '$NB' " " '$NB' " " '$NB'}{print}' \
tmp4.txt  > tmp5.txt

awk 'BEGIN{print "%%MatrixMarket matrix coordinate \
 real general"}{print}' tmp5.txt   > $2

rm tmp.txt tmp2.txt tmp3.txt tmp4.txt tmp5.txt
```

6.4 Plot by Graphic User Interface

Silly daemons