EDISON

Electromagnetic Design of
flexIble SensOrs

FNP Team-Tech

Raport 84 - SLEPC, PETSC

dr eng. Grzegorz Fotyga
July 2, 2020

European Funds Smart Growth

FNP Foundation for Polish Science

European Union
European Regional Development Fund

The „EDISOn - Electromagnetic Design of flexIble SensOrs" project, agreement no TEAM TECH/2016-1/6, is carried out within the TEAM-TECH programme of the Foundation for Polish Science co-financed by the European Union under the European Regional Development Fund.

1 Configuration (SLPC+PETSC)

1. SLEPC version should be the same as PETSC, otherwise it is not going to work.
2. Install:
   (a) Cygwin (X64 version used in this report) Packages for Cygwin:
      - gcc,g++
      - Python (preferred version is 2.7, the version 3 and newer versions DO NOT WORK, because is not supported by PETCS yet).
      - make
      - flex
      - git

   (b) Visual Studio
3. After inslatlling Cygwin64, open the `Cygwin.bat` with text editor and add
   `CALL "\the\path\to\vcvarsall.bat"amd64`
   For example:
   `CALL "C:\ProgramFiles\VisualStudio\vcvarsall.bat" amd64`
   Vcralsall.bat is pointing to native command prompt for Microsoft Visual Studio for 64-bit version
4. Press the windows key on keyboard and search for Native command prompt for X64, run it as an Administrator account
5. Go to Cygwin64 directory and lunch Cygwin.bat
6. Go to `C:\`, untar the PETSC using command `tar -xzf petsc-3.12.2.tar.gz`
7. Go to PETSC directory, configure it using:

   - **(32 bits!)**
     ```
     ./configure --CFLAGS=" -I/cygdrive/c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl/include " -
     " -I/cygdrive/c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl/include " --COPTFLAGS=
     " -Ofast -xHost" --CXXOPTFLAGS=" -Ofast -xHost" --with-cc='win32fe cl' --with-cxx=
     'win32fe cl' --with-fc=0 --with-mpi=0 --with-shared-libraries=1 --with-64-bit-indices
     0 --with-blaslapack-lib='-L/cygdrive/c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl/
     lib/intel64 mkl_intel_lp64_dll.lib mkl_sequential_dll.lib mkl_core_dll.lib' --with-mkl
     sparse=1 --with-64-bit-blas-indices=0 --known-64-bit-blas-indices=0 --with-mkl_
     sparse-dir='/cygdrive/c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl' --with-mkl_pardiso=
     1 --with-mkl_pardiso_dir='/cygdrive/c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl/
     lib/intel64' --with-mkl_sparse_optimize=1 --with-mkl_sparse_optimize-dir='/cygdrive/
     c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl' --with-mkl_pardiso-include="/cygdrive/
     c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl/include"
     ```
   - **(64 bits, no pardiso!)**
     ```
     ./configure --CFLAGS=" -I/cygdrive/c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl/include " -
     " -I/cygdrive/c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl/include " --COPTFLAGS=
     " -Ofast -xHost" --CXXOPTFLAGS=" -Ofast -xHost" --with-cc='win32fe cl' --with-cxx=
     'win32fe cl' --with-fc=0 --with-mpi=0 --with-shared-libraries=1 --with-64-bit-indices
     1 --with-blaslapack-lib='-L/cygdrive/c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl/
     lib/intel64 mkl_intel_lp64_dll.lib mkl_sequential_dll.lib mkl_core_dll.lib'
     ```
   - **(32 bits, pardiso, complex!)**
     ```
     ./configure --CFLAGS=" -I/cygdrive/c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl/include " -
     " -I/cygdrive/c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl/include " --COPTFLAGS=
     " -Ofast -xHost" --CXXOPTFLAGS=" -Ofast -xHost" --with-cc='win32fe cl' --with-cxx=
     'win32fe cl' --with-fc=0 --with-mpi=0 --with-shared-libraries=1 --with-64-bit-indices
     ```

```
0 --with-blaslapack-lib='-L/cygdrive/c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl/
lib/intel64 mkl_intel_lp64_dll.lib mkl_sequential_dll.lib mkl_core_dll.lib' --with-mkl
sparse=1 --with-64-bit-blas-indices=0 --known-64-bit-blas-indices=0 --with-mkl_
sparse-dir='/cygdrive/c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl' --with-mkl_pardiso=
1 --with-mkl_pardiso_dir='/cygdrive/c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl/
lib/intel64' --with-mkl_sparse_optimize=1 --with-mkl_sparse_optimize-dir='/cygdrive/
c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl' --with-mkl_pardiso-include="/cygdrive/
c/PROGRA~2/INTELS~1/COMPIL~2/windows/mkl/include"  --with-scalar-type=complex --with-c
cxx
```

8. After getting successfully configured, you will see a message that the next step is "make debug all" and "make check".
9. Go to the PETSC directory and run export
   - export PETSC_DIR=$PWD
   - export PETSC_ARCH=arch-mswin-c-debug (sometimes cxx!)
10. Untar SLEPC, go to slepc directory, export the path:
    - export SLEPC_DIR=$PWD
11. Run *./configure*, than *make* and *check*, following the instructions in the command line.

2 Create a standalone project

1. Generate standard VS console project, (cpp code is provided here)
2. Copy MKL dll's to project folder (with project.exe) from `C:\Program Files (x86)\IntelSWTools\compilers_and_libraries_2019.4.245\windows\redist\intel64_win\mkl`
3. Copy `libslepc.dll` and `libpetsc.dll` from `C:\petsc-3.12.2\arch-mswin-c-debug\lib` and `C:\slepc-3.12.2\arch-mswin-c-debug\lib` to the same folder.
4. Add in the project the following options in C++ -> general (with proper slepc/petsc versions, remember to set the proper debug, x64 options in VS window):
   `C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.21.27702\crt\src\vcruntime \\ C:\slepc-3.12.1\src\\ C:\slepc-3.12.1\arch-mswin-c-debug\include\\ C:\petsc-3.12.1\src\\ C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.21.27702\crt\src\concrt\\ C:\slepc-3.12.1\include\\ C:\petsc-3.12.1\arch-mswin-c-debug\include\\ C:\petsc-3.12.1\include`
5. Add in the project the following options in Linker -> input data (with proper slepc/petsc versions)
   `C:\slepc-3.12.1\arch-mswin-c-debug\lib\libslepc.lib C:\petsc-3.12.1\arch-mswin-c-debug\lib\libpetsc.lib`

3 Run a standalone SLEPC project

The goal is to run a standalone project that computes the eigenvalues i.e. resonant frequencies of the resonant cavity of the electromagnetic problem in 64 bit.

1. Export Tt and St matrices from the InventSim project `simple_waveguide_eigs`
2. Put Tt and St into path `C:\Users\GFotyga\Source\Repos\hello_petsc\x64\Debug\convert_matrices64`
3. run `LoadData_All.m` to obtain Tt and St in matlab

4. Now, we have to use the matlab tool to export the Matlab matrices to the PETSC format. To this end, we will use the tool available in `https://github.com/erdc/petsc-dev/tree/master/bin/matlab`. Firstly, we have to change `indices = 'int32';` to `indices = 'int64';` in `PetscBinaryWrite`

5. Export Tt and St to PETS format: `PetscBinaryWrite('C:\Users\GFotyga\Source\Repos\hello_petsc\x64\Debug\convert_matrices\Tt_150k_64bit.petsc', Tt)` and similarly to `St_150k_64bit.petsc` matrix. Now the matrices are stored in 64 bit PETSC format.

6. Copy the PETSC matrices `Tt_150k_64bit.petsc` and `St_150k_64bit.petsc` to the standalone project (configured in section I), that computes the resonant frequencies of the electromagnetic problem. `C:\Users\GFotyga\Source\Repos\slepc_with_mkl\x64\Debug`

7. Now, run the `matlab_driver C:\Users\GFotyga\Source\Repos\slepc_with_mkl\x64\Debug`