Installation Guide for Windows 10

Step 0. Set environment

Here the procedure is demonstrated using

`Windows 10 + git-2.34 + cmake-3.22 + Visual Studio 2022 + Intel oneAPI/oneMKL 2022`

In the Intel CMD for x64, the environment is usually set by default. In case you would like to use a more powerful Unix-like shell tool **PowerShell**, you can run

```
powershell
```

If the environment are not set, first set necessary variables by hand with:

```
"C:\Program Files (x86)\Intel\oneAPI\setvars.bat"
```

If you do not do anything, your system may use MSVC instead of Intel compilers. In case you want to use the Intel 2022 compilers (icl/icx/dpcpp), you can use the following (icl for Intel Classical C++, icx for Intel NextGen C++, dpcpp for Intel DPC++):

```
set CC=icx
set CXX=icx
```

If you are using **PowerShell** instead of the regular command line tool **cmd**, set the environment varilabes with

```
$env:CC='icx'
$env:CXX='icx'
```

Step 1. Download and install faspsolver (required)

The solver package faspsolver is currently required. You can get it from the GitHub repository

`git@github.com:FaspDevTeam/faspsolver.git`

After obtained the package, run:

```
cd faspsolver; mkdir Build; cd Build
```

Note: The **;** notation only works in **PowerShell**; if you use plain **cmd**, you need to run three commands one by one!

Now you can generate a VS2022 solution with MKL Pardiso support

```
cmake -T "Intel C++ Compiler 2022" -DCMAKE_C_COMPILER="icx" -DUSE_PARDISO=ON ..
```

Here we assume there is only one Visual Studio available; and, it will be used by default.

If you have multiple VS versions (for example, VS2022 and VS2019) installed on your system, you can use specify which VS to use using the **-G** option. For example, in order to use VS2022, you may run

```
cmake -G "Visual Studio 17 2022"
```

Once cmake successed, open faspsolver.sln and build the **ALL_BUILD** target as well as the **INSTALL** target.

Step 2. Download and install fasp4blkoil (optional)

If case you only wish to use the basic solvers in *faspsolver*, you can skip this step. In case you want to use the preconditioners from *fasp4blkoil*, you should first download it from

`git@github.com:FaspDevTeam/fasp4blkoil.git`

Similar to the above steps, run

```
cd fasp4blkoil
mkdir Build
cd Build
```

Then you need to tell **cmake** where is faspsolver (replace the dir name with your setting) by

```
set FASP_DIR=\prog\0.FASP\faspsolver
```

Again, if you are using **PowerShell**, just use

```
$env:FASP_DIR='\prog\0.FASP\faspsolver'
```

Generate VS2022 solution with the MKL Pardiso support with

```
cmake -T "Intel C++ Compiler 2022" -DCMAKE_C_COMPILER="icx" -DUSE_PARDISO=ON ..
```

Open fasp4blkoil.sln and build the **ALL_BUILD** and **INSTALL** targets.

Usually, the above two steps are only needed for the first time.

Step 3. Download and install OpenCAEPoro

Now we are ready to build *OpenCAEPoro*. First, download it from

`git@github.com:FaspDevTeam/OpenCAEPoro.git`

Then follow the standard steps to generate Visual Studio solutions using cmake:

```
cd OpenCAEPoro
mkdir Build
cd Build
set FASP_DIR=\prog\0.FASP\faspsolver
set FASP4BLKOIL_DIR=\prog\0.FASP\fasp4blkoil
```

If you use **PowerShell** instead of the regular command line tool **cmd**, set the environment varilabes with

```
cd OpenCAEPoro; mkdir Build; cd Build
$env:FASP_DIR='\prog\0.FASP\faspsolver'
$env:FASP4BLKOIL_DIR='\prog\0.FASP\fasp4blkoil'
```

After setting the environment, you can run (if you have fasp4blkoil and intel MKL)

```
cmake -T "Intel C++ Compiler 2022" -DCMAKE_C_COMPILER="icx" -DUSE_FASP4BLKOIL=ON -
DUSE_PARDISO=ON ..
```

or just simply

```
cmake -T "Intel C++ Compiler 2022" -DCMAKE_C_COMPILER="icx" ..
```

Now you are ready. Just open OpenCAEPoro.sln and build the **ALL_BUILD** target. And then build **INSTALL** target if you wish to install the lib and exe files to desirable directories.

In case you wish to generate the **DEBUG** solution for debugging your code, you may use

```
cmake -T "Intel C++ Compiler 2022" -DCMAKE_C_COMPILER="icx" -
DCMAKE_BUILD_TYPE=Debug ..
```

Modified on March/08/2022 by Chensong Zhang