OPENWARP - PROVIDE A COMMAND LINE INTERFACE USING PYTHON DOCUMENTATION

TCSASSEMBLER, 30/05/2016

1- Implementation Details

a) Submission files

Added:

source/openwarpgui/configs/default_configuration.json This contains the default configuration
source/openwarpgui/openwarp_cli.py This is the main OpenWarp cli interface

The above are testing configurations file
source/openwarpgui/test_files/configs/2bodies.json
source/openwarpgui/test_files/configs/cylinder.json
source/openwarpgui/test_files/configs/flapsingle_igs.json
source/openwarpgui/test_files/configs/multiple_sim_same_mesh.json
source/openwarpgui/test_files/configs/same_conf_diff_geometry.json
source/openwarpgui/test_files/configs/shell_step.json

For modified files list check the patch

b) Why the configuration was provided as JSON

OpenWarp GUI is actually a wrapper around Nemoh. And the previous version of Nemoh (the current too as we did not remove this feature) had command line support together with full support of Nemoh.cal file as configuration. But, some features and configurations available in Nemoh were not configurable through Nemoh.cal.

We could have extended Nemoh.cal to handle those additional features, however Nemoh.cal is a line based format and achieving the client use case with it will not be robust. Indeed, the order of the configuration must be respected and adding the wrong character or adding a new line to the file or the wrong space will make the configuration become invalid. Finally it would not be intuitive to run multiple simulations from the same Nemoh.cal.

On the other hand Nemoh fully supported another configuration file in the format HDF5. Actually the native format recognize by Nemoh is that HDF5 format. And Nemoh will write all important output and results into a file named db.hdf5. This HDF5 format is a good candidate for configuration especially given that it was fully supported and can solve the above issues very well. However it is not a text file and don't match the contest requirement. Furthermore it is very difficult to edit without coding.

This is why we implemented a new format and used JSON to solve the aforementioned issues.

The format has property name intuitive to the user. Also it is possible to configure multiple simulations in the same configuration file without having to duplicate them. This is done by supporting some default values.

Furthermore, the user can have multiple simulations and specify which one to run

2- Usage

Before being able to run the cli, you should setup your environment library path to contain the Nemoh fortran library directory.

Note that the setup of the environment variables was done in the one-click installers in the script run.sh run.command and run.bat respectively for linux, OS X and Windows. Those files are available from the installers. And they run directly the GUI after setting up the environment variables.

We suggest that when the one-click installers are rebuilt, they include such facility scripts for running the CLI too. In the meantime, one can source the existing script to run the GUI and then quit them using Control + C before starting the CLI. To source the script use:

source run.sh ; **source run.command** ; **call run.bat** respectively for Linux, OS X and Windows

To use the openwork cli, simply run: **python openwarp_cli.py configuration1 …**

where configuration* is the path to the configuration file or a JSON string

The configuration fields follow an intuitive name easy to identify and comparable to the one in the GUI screens.

There is a “**phase**” field. Set it to "MESHING" to only do the meshing step.
Set it to "POSTPROCESSING" to only do the postprocessing step, set it to "SIMULATION" to only do the simulation step. When set to None or empty array, all the steps are automatically done.

You can define multiple simulations in the configurations by specifying a different ID for each one. To avoid repeating the same settings, you can specify common settings in the simulations with id “**default**”. Similarly, within in each simulation, you can defined multiple bodies with different ID for each one. Common bodies settings can be defined with the body with id “**default**”.

You can specify which simulations to run by configuring the field: "**simulations_to_run**". It should an array of simulations ID. Note that the "**default**" ID is reserved and is not run. It is only used to provide default parameters. If you set "**simulations_to_run**" to None or empty array, all simulations are run.

You can set the "**verbosity**" setting to 0 to disable most logging output from the terminal screen. Note that all logs would still be added to the logging file. Set the "**verbosity**" to any integer higher than 0 to show all output also on screen.

3- Verification

The simplest way to test is by using Linux:

- Clone the git repository https://github.com/cloudspokes/OpenWARP and switch to branch 30054369_cli using for example the command: **git clone -b 30054369_cli --single-branch https://github.com/cloudspokes/OpenWARP.git**
- Apply the patch file from our submission to get the new code using **git apply --whitespace=nowarn patch_file.patch** by replacing patch_file.patch with the location of the patch file.
- Download the linux installer from https://github.com/cloudspokes/OpenWARP/tree/master/installers
- Extract it and update the installer files with the file of the new code (the source directory after applying the patch)
- Run the file installer.sh with sudo

For OSX or Windows user, note that we added 1 new requirement for Python. So it is important to install them as explained in the deployment guide.

We tested on OSX. We make sure that the configuration file can be run with no path issue from the directory containing openwarp_cli.py. However if there is any path issue (most likely, uppercase, lowercase one), feel free to update the path in the respective configuration.

For all test cases cross checked that at least the input values inside the db.hdf5 match the configuration given in the JSON file. The db.hdf5 file will be located inside the tested simulation directory. All simulation directories are sub-directory of source/openwarpgui/test_files/cli_results

a) Basic Testing

Run **python openwarp_cli.py test_files/configs/cylinder.json**

Once finished, you should see something like :

```
2016/05/30 17:50:58            INFO Problem   208 /   280 ... Done !
2016/05/30 17:50:58            INFO Problem   173 /   280 ... Done !
2016/05/30 17:50:58            INFO Problem   139 /   280 ... Done !
2016/05/30 17:50:59            INFO Problem   244 /   280 ... Done !
2016/05/30 17:50:59            INFO Problem   279 /   280 ... Done !
2016/05/30 17:50:59            INFO Problem    69 /   280 ... Done !
2016/05/30 17:50:59            INFO Problem   104 /   280 ... Done !
2016/05/30 17:50:59            INFO Problem    34 /   280 ... Done !
2016/05/30 17:50:59            INFO Problem   209 /   280 ... Done !
2016/05/30 17:50:59            INFO Problem   174 /   280 ... Done !
2016/05/30 17:50:59            INFO Problem   140 /   280 ... Done !
2016/05/30 17:50:59            INFO Problem   245 /   280 ... Done !
2016/05/30 17:50:59            INFO Problem   280 /   280 ... Done !
2016/05/30 17:50:59            INFO Problem    70 /   280 ... Done !
2016/05/30 17:50:59            INFO Problem   105 /   280 ... Done !
2016/05/30 17:50:59            INFO Problem    35 /   280 ... Done !
2016/05/30 17:50:59            INFO Problem   210 /   280 ... Done !
2016/05/30 17:50:59            INFO Problem   175 /   280 ... Done !
 2016/05/30 17:50:59             INFO NEMOH Solver completed.
/Applications/OpenWarp/openwarpgui/test_files/cli_results/cylinder/results/radiationcoefficients.tec contains t
he radiation coefficients in tec format.

/Applications/OpenWarp/openwarpgui/test_files/cli_results/cylinder/results/diffractionforce.tec contains the di
ffraction forces in tec format.

/Applications/OpenWarp/openwarpgui/test_files/cli_results/cylinder/results/excitationforce.tec contains the exc
itation forces in tec format.

/Applications/OpenWarp/openwarpgui/test_files/cli_results/cylinder/results/irf.tec contains the irf in tec form
at.

/Applications/OpenWarp/openwarpgui/test_files/cli_results/cylinder/results/WaveField.tec contains the wave elev
ation in tec format.

The post processing results are saved in the hdf5 file /Applications/OpenWarp/openwarpgui/test_files/cli_result
s/cylinder/db.hdf5
```

This test is just to make sure that all the configurations are working correctly

b) shell_step.json

This test is to make sure that the command line will automatically generate a mesh file in the dat format without the user needing to specify it. Indeed, Nemoh only support DAT_FILE. And in the GUI, one can use the MESHING screen to convert a file in (shell, stl, igs) to dat file before giving it as input in the SIMULATION SCREEN.
The command line interface simplify this and make do this conversion when needed.

After a successful run of this test you should see a directory meshing inside the simulation directory with the dat created and used. Something like:

```
├── db.hdf5
├── mesh
│   └── mesh.tec
├── meshing
│   ├── config.txt
│   ├── log.txt
│   ├── shell-quad.dat
│   ├── shell-quad.gdf
│   ├── shell-quad.stl
│   ├── shell-quad.vtk
│   ├── shell-quad.vtp
│   ├── shell.gdf
│   ├── shell.stl
│   ├── shell.vtk
│   ├── shell.vtp
│   └── test.out
├── postprocessing_log.txt
├── results
│   ├── WaveField.tec
│   ├── diffractionforce.tec
│   ├── excitationforce.tec
│   ├── fkforce.tec
│   ├── irf.tec
│   └── radiationcoefficients.tec
└── simulation_log.txt

3 directories, 22 files
```

c) multiple_sim_same_mesh.json

This is to test that we can run multiple different simulations with the same mesh file.

Make sure the simulations defined here are generated

d) same_conf_diff_geometry.json

This is to test that we can run multiple different simulations with the same configurations on different geometry.

e) flapsingle_igs.json

This is to test the flapsingle geometry.

This will take an awfully lot of time (More that 10h) to complete and require a lot of memory.

f) Error Message

When the configuration is invalid or the command misused an appropriate error message is shown. Check screen shot:

```
Aristides-MacBook-Pro:openwarpgui yedtoss$ python openwarp_cli.py
Found logging configuration file at /Applications/OpenWarpGood/openwarpgui/nemoh
/logging.json

Writing log at /Users/yedtoss/OpenWarpFiles/logs/logs.log

running build_ext
Error: No configurations file given to the CLI. Usage of script is: openwarp_cli
 configuration1 ..

Aristides-MacBook-Pro:openwarpgui yedtoss$ python openwarp_cli.py /path/not/foun
d
Found logging configuration file at /Applications/OpenWarpGood/openwarpgui/nemoh
/logging.json

Writing log at /Users/yedtoss/OpenWarpFiles/logs/logs.log

running build_ext
Error: Configuration file/json /path/not/found not found/not valid

Aristides-MacBook-Pro:openwarpgui yedtoss$ python openwarp_cli.py '{".invalid_js
on_string}'
Found logging configuration file at /Applications/OpenWarpGood/openwarpgui/nemoh
/logging.json

Writing log at /Users/yedtoss/OpenWarpFiles/logs/logs.log

running build_ext
Error: Configuration file/json {".invalid_json_string} not found/not valid

Aristides-MacBook-Pro:openwarpgui yedtoss$
```