

Deployment Guide

yedtoss

September 26, 2016

Contents

1	INTRODUCTION	2
2	OVERVIEW	2
3	NOTATIONS	2
4	DEPENDENCIES	3
4.1	Windows	3
4.2	OS X	3
4.3	Linux	4
5	Building from scratch	4
5.1	WINDOWS DEPLOYMENT FROM SCRATCH	4
5.2	OS X DEPLOYMENT FROM SCRATCH	6
5.3	Linux deployment from scratch	16
6	Using Install Script	16
6.1	Windows	16
6.2	OS X	17
6.3	Linux	18
7	Deployment from existing package	19
7.1	Windows	19
7.1.1	Recompiling NemohFortran and NemohPython	19
7.1.2	Repackaging the installer	19
7.2	OS X	19
7.2.1	Recompiling NemohFortran and NemohPython	19
7.2.2	Packaging the installer	20
7.3	Linux	20
7.3.1	Recompiling NemohFortran and NemohPython	20
7.3.2	Packaging the installer	20
7.4	General Guide about the installer build/packaging procedure	21

8 Configuration	21
8.1 Meshing parameters	21
8.2 Simulation parameters	22
8.3 Postprocessing parameters	24
8.4 Other and Logging parameters	24
9 Usage	24
9.1 Start the application	24
9.2 Start meshing	26
9.3 Execute meshing	27
9.4 View the meshing results	27
9.5 Start the simulation	28
9.6 Execute the simulation	29
9.7 View the simulation results	29
9.8 Postprocessing	31
9.9 Generate TEC outputs	32
9.10 Postprocessing results	33
9.11 Visualize the generated TEC files	33
9.12 Getting help	34
9.13 Logging Support	34
10 Python CLI	37
10.1 CLI Verification	37
11 Automated TestScript	38
11.1 Repository	38
11.2 Dependencies	38
11.3 Test 1: Converting GDF to OpenWarp Format	39
11.3.1 Objective of Test case	39
11.3.2 Steps to execute the test	39
11.3.3 Output	39
11.4 Test 2: Wamit Test Input and Output	39
11.4.1 Objective of Test case	39
11.4.2 Steps to execute the test	39
11.4.3 Output and Comparison	39
11.5 Test 3: Testing HigherOrder Implementation in OpenWarp	40
11.5.1 Objective of Test case	40
11.5.2 Steps to execute the test	40
11.5.3 Output and Comparison	40
11.6 Test 4: Testing the Dipoles/Thin Panels implementation in OpenWarp	40
11.6.1 Objective of Test case	40
11.6.2 Steps to execute the test	40
11.6.3 Output and Comparison	40
11.7 Test 5: Testing Irregular Frequency Removal in OpenWarp	40
11.7.1 Objective of Test case	40

11.7.2 Steps to execute the test	40
11.7.3 Output and Comparison	40
11.8 Test 6: Testing Mean Drift Forces and Yaw moment Implementation in Open-Warp	41
11.8.1 Objective of Test case	41
11.8.2 Steps to execute the test	41
11.8.3 Output and Comparison	41
11.9 Test 7: Computation of the Green Function using ordinary differential equation in OpenWarp	41
11.9.1 Objective of Test case	41
11.9.2 Steps to execute the test	41
11.9.3 Output and Comparison	41
11.10 Test 8: Testing Green Function Tabulation Implementation	41
11.10.1 Objective of Test case	41
11.10.2 Steps to execute the test	41
11.10.3 Output and Comparison	41
11.11 Test 9: 2Bodies	42
11.11.1 Objective of Test case	42
11.11.2 Steps to execute the test	42
11.11.3 Output and Comparison	42
11.12 Test 10: Cylinder	42
11.12.1 Objective of Test case	42
11.12.2 Steps to execute the test	42
11.12.3 Output and Comparison	42
11.13 Test 11: NonSymmetrical	42
11.13.1 Objective of Test case	42
11.13.2 Steps to execute the test	42
11.13.3 Output and Comparison	42
12 Steps to update OW from Nemoh 2014 to Nemoh 2016	43
13 Practical Example	43
13.1 Mesh Generation	43
13.2 Run the simulation	45
13.3 Generate TEC files and run postprocessing	47
13.4 Visualize the generated tec files	48

1 INTRODUCTION

OpenWarp is a web application that can mesh and simulate a body of an offshore structure. It computes the first-order forces (added mass, radiation damping, and diffraction forces) on the body and solves the radiation-diffraction problem of first order.

The OpenWarp software package is composed of three main parts. The first part is the meshing tool. The tool can take as input a raw body in STEP, STL, or IGS format and generate its corresponding shell. The second part of OpenWarp, called Nemoh Solver, is an application that solves the problem of seakeeping in hydrodynamics. It uses the boundary element method to solve for bodies partially immersed (floating) or completely immersed in a fluid of infinite or constant finite depth, with or without forward speed subject to sinusoidal waves. It can be used with the first-order or higher-order panel method. The third part postprocesses the results obtained by Nemoh Solver and generates TECPLOT files of diffraction, radiation, and excitation forces. Added mass and damping coefficients are also generated in TECPLOT format.

OpenWarp operates on an input mesh. To get the input mesh in the correct format, one should use the meshing tool, which can generate a thin or non-thin mesh from an existing body described in STEP, STL, or IGS format. To solve the radiation and diffraction problem, we must represent the body with multiple (thin) quadrilateral elements or shells. OpenWarp includes a web application which can be used to mesh, simulate, and postprocess a body. This web application has a graphical interface that is powered by a server which the user runs locally.

OpenWarp has been developed by the Department of Energy and Topcoder. It is expected to be used by people developing Wave Energy Conversion (WEC) devices in the United States. Until now the testing of new WEC devices has been expensive and time-consuming. OpenWarp will spur innovation by enabling wave-energy startups to develop, analyze, and optimize their devices more quickly.

2 OVERVIEW

Immediately below, we list the notations used in this document and describe the software dependencies for OpenWarp deployment. In subsequent sections, we give detailed instructions for deploying OpenWarp on Windows and on OS X. Finally, we describe how to configure OpenWarp and give a practical usage example

3 NOTATIONS

The following notations are used in this document.

\$ROOT : the top-level directory of the Nemoh software package.

\$NEMOH_FORTRAN : the directory **\$ROOT/NemohImproved/Nemoh/**

\$FORTRAN_BUILD : the build directory for the FORTRAN version of Nemoh.

\$MINGW_ROOT : the directory where MinGW will be installed.

\$INSTALL_DIR is used in the following to mean the root directory containing the files you get when you install the currently available installer for each os.

\$NEMOH_PYTHON : the directory **\$ROOT/openwarpgui/nemoh**

4 DEPENDENCIES

OpenWarp is compatible with Windows and OS X. Below are the software packages needed to install the application. Section 5 provides step-by-step instructions for Windows installation and section 6 does likewise for OS X.

4.1 Windows

- **Windows 7/8 64 bits** (Other versions might work but not tested)
- MinGW 4.8.1 <http://sourceforge.net/projects/MingWbuilds/files/host-windows/releases/4.8.1/>
- BLAS <http://icl.cs.utk.edu/lapack-for-windows/lapack/#libraries>
- LAPACK <http://icl.cs.utk.edu/lapack-for-windows/lapack/#libraries>
- OpenMP provided by MinGW
- HDF5 >= 1.8.11 <http://www.hdfgroup.org/HDF5/> Optionally provided by Anaconda
- HDFView <http://www.hdfgroup.org/products/java/release/download.html> Python 2.7 Optionally provided by Anaconda
- H5py >= 2.3.1 Optionally provided by Anaconda
- NumPy Optionally provided by Anaconda
- CMake >= 2.8 <http://www.cmake.org/cmake/resources/software.html>
- Anaconda (with Python 2.7) >= 2.1.0 <http://continuum.io/downloads>
- ParaView >= 4.1 <http://www.paraview.org/download/>

4.2 OS X

- **MAC OS X 10.10 64 bits** (Other versions might work but not tested)
 - Homebrew (brew command) <http://brew.sh/>
 - GCC >= 4.8 installed by brew
 - Xcode Command Line Tools installed by brew
-

- BLAS provided by XCode commands
- LAPACK Provided by XCode commands
- OpenMP provided by GCC
- HDF5 $\geq 1.8.11$ <http://www.hdfgroup.org/HDF5/> Optionally provided by Anaconda
- HDFView <http://www.hdfgroup.org/products/java/release/download.html>
- Python 2.7 Optionally provided by Anaconda
- H5py $\geq 2.3.1$ Optionally provided by Anaconda
- NumPy Optionally provided by Anaconda
- CMake ≥ 2.8 installed by brew
- Anaconda (with Python2.7) $\geq 2.1.0$ <http://continuum.io/downloads>
- ParaView ≥ 4.1 <http://www.paraview.org/download/>

4.3 Linux

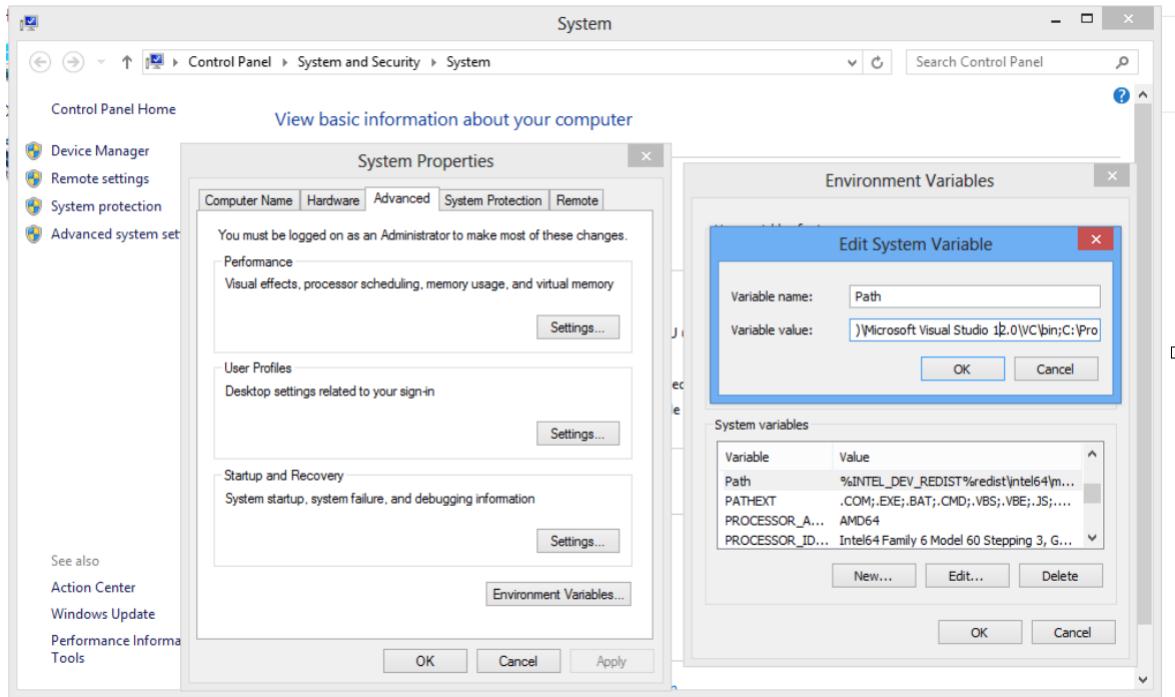
- Linux/Ubuntu 14.04 64 bits (Other versions might work but not tested)
- Except from Homebrew and XCode, Linux shared the same dependencies as MAC OSX as listed in section 4.2.

5 Building from scratch

5.1 WINDOWS DEPLOYMENT FROM SCRATCH

Only 64-bit versions of Windows 7 and Windows 8 are supported. To deploy the application on Windows, perform the following steps:

- Download MinGW (64-bit) from <http://sourceforge.net/projects/mingwbuilds/files/host-windows/releases/4.8.1/64-bit/threads-posix/sjlj/x64-4.8.1-release-posix-sjlj-rev5.7z/download>.
- Extract it to a directory, making sure that the directory name contains no spaces. Let's denote this directory as **\$MINGW_ROOT**.
- Add **\$MINGW_ROOT\bin** and **\$MINGW_ROOT\lib** to the beginning of your Windows PATH by opening the System control panel and editing Properties → Advanced System Settings → Advanced tab → Environment Variables → Path as shown in this screenshot:



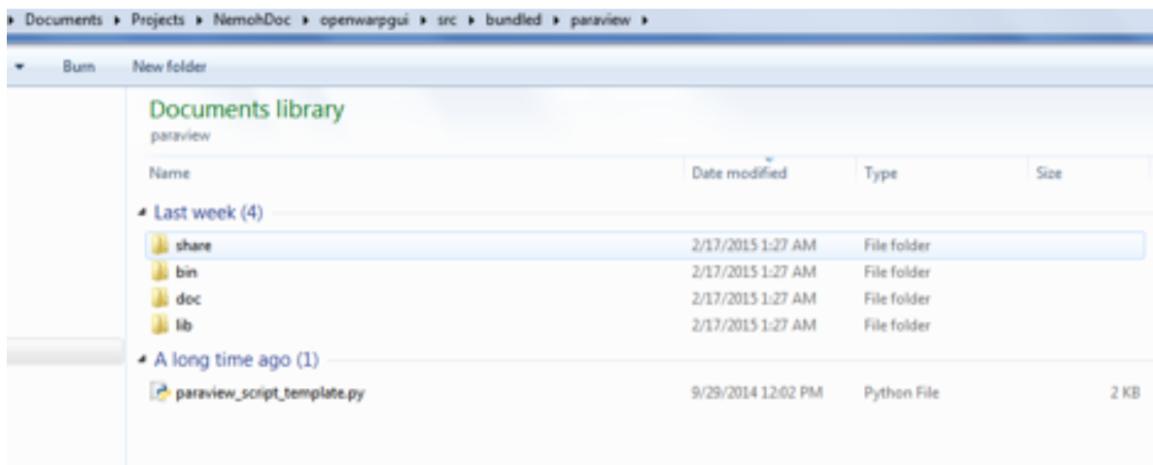
- Copy `$ROOT/src/bundled/simulation/libs/libnemoh.dll`, `$ROOT/src/bundled/simulation/libs/libnemoh.dll.a`, `$ROOT/src/bundled/simulation/libs/libblas.dll`, and `$ROOT/src/bundled/simulation/libs/liblapack.dll` to `$MINGW_ROOT/lib`.
- Download and install Anaconda 2.10 with Python 2.7 for Windows (64-bit, graphical installer) from <http://continuum.io/downloads>. If you install it to `C:/Users/bob/Anaconda` for example, make sure that you manually add `C:/Users/bob/Anaconda` and `C:/Users/bob/Anaconda/Scripts` to the beginning of your Windows PATH variable. Don't forget to replace bob with your own user name.
- Once the preceding steps are done, start PowerShell and install the CherryPy dependencies by executing this command in the `$ROOT/src/` directory: `pip install -r requirements.txt`

```
PS C:\Users\yedtoss> cd C:\Users\yedtoss\Documents\Projects\NemohDoc\openwarpgui\src
PS C:\Users\yedtoss\Documents\Projects\NemohDoc\openwarpgui\src> pip install -r requirements.txt
Requirement already satisfied (use --upgrade to upgrade): CherryPy in c:\users\yedtoss\anaconda\lib\site-packages (from -r requirements.txt (line 1))
Requirement already satisfied (use --upgrade to upgrade): numpy in c:\users\yedtoss\anaconda\lib\site-packages (from -r requirements.txt (line 2))
Requirement already satisfied (use --upgrade to upgrade): cython in c:\users\yedtoss\anaconda\lib\site-packages (from -r requirements.txt (line 3))
Requirement already satisfied (use --upgrade to upgrade): h5py in c:\users\yedtoss\anaconda\lib\site-packages (from -r requirements.txt (line 4))
Cleaning up...
PS C:\Users\yedtoss\Documents\Projects\NemohDoc\openwarpgui\src>
```

- Start the server by executing this in the `src/` directory: `python main.py`

```
[24/Feb/2015:16:27:33] ENGINE Bus ENTRIES
PS C:\Users\yedtoss\Documents\Projects\NemohDoc\openwarpgui\src> python .\main.py
running build_ext
[24/Feb/2015:16:27:33] ENGINE Listening for SIGTERM.
[24/Feb/2015:16:27:33] ENGINE Bus STARTING
[24/Feb/2015:16:27:33] ENGINE Set handler for console events.
[24/Feb/2015:16:27:33] ENGINE Started monitor thread '_TimeoutMonitor'.
[24/Feb/2015:16:27:33] ENGINE Started monitor thread 'Autoreloader'.
[24/Feb/2015:16:27:33] ENGINE Serving on http://127.0.0.1
[24/Feb/2015:16:27:33] ENGINE Bus STARTED
```

- Optionally, download the .zip version of ParaView from <http://www.paraview.org/download/> and copy it to src/bundled/paraview so that src/bundled/paraview/bin exists. This is only needed for testing the visualization of the results in ParaView.



- If the provided libnemoh.dll and libnemoh.dll.a do not work, you can generate them manually by doing the following:
 - Download CMake from <http://www.cmake.org/files/v2.8/cmake-2.8.12.2-win32-x86.exe>, install it and make sure it is in your PATH variable.
 - Start PowerShell and enter an empty directory. Then run:
 - `cmake -DCMAKE_Fortran_COMPILER="gfortran" "$NEMOH_FORTTRAN" -G "MinGW Makefiles"` And finally run: `mingw32-make`

5.2 OS X DEPLOYMENT FROM SCRATCH

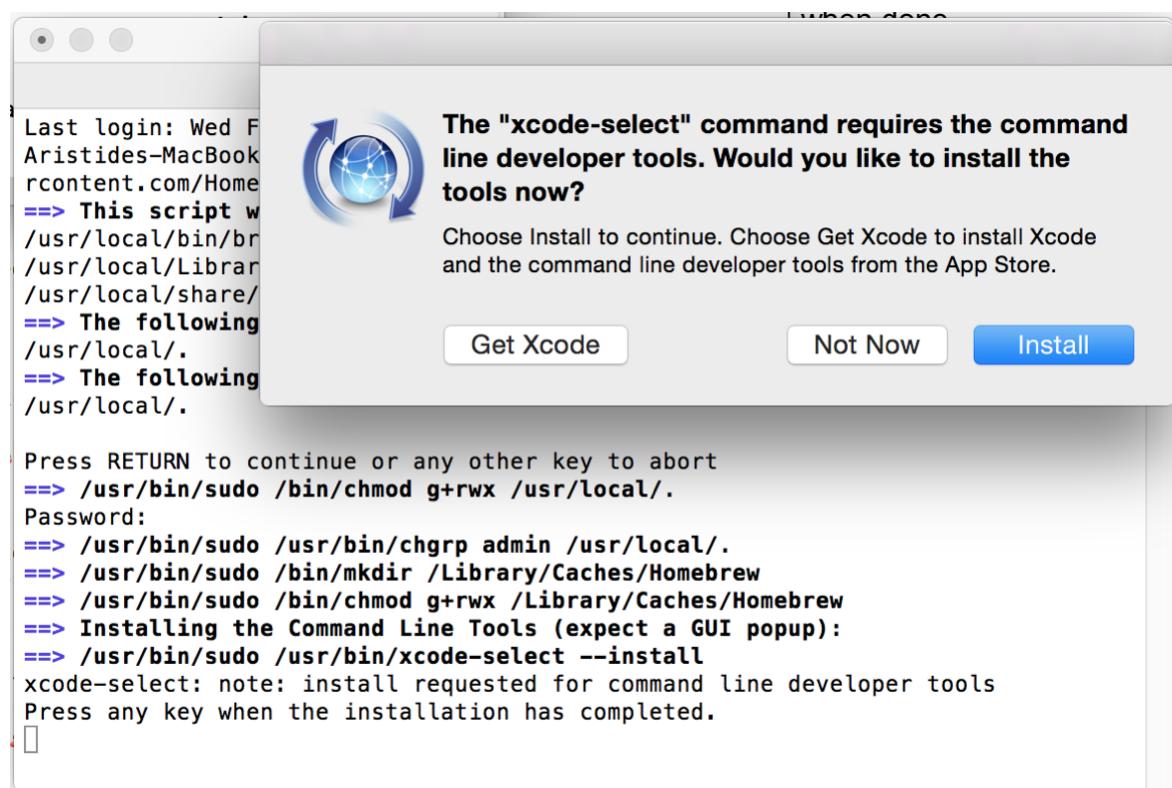
The following instructions are known to work on OS X versions 10.9 and 10.10. To deploy the application in OS X, perform the following step:

- Install brew: `ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"`.

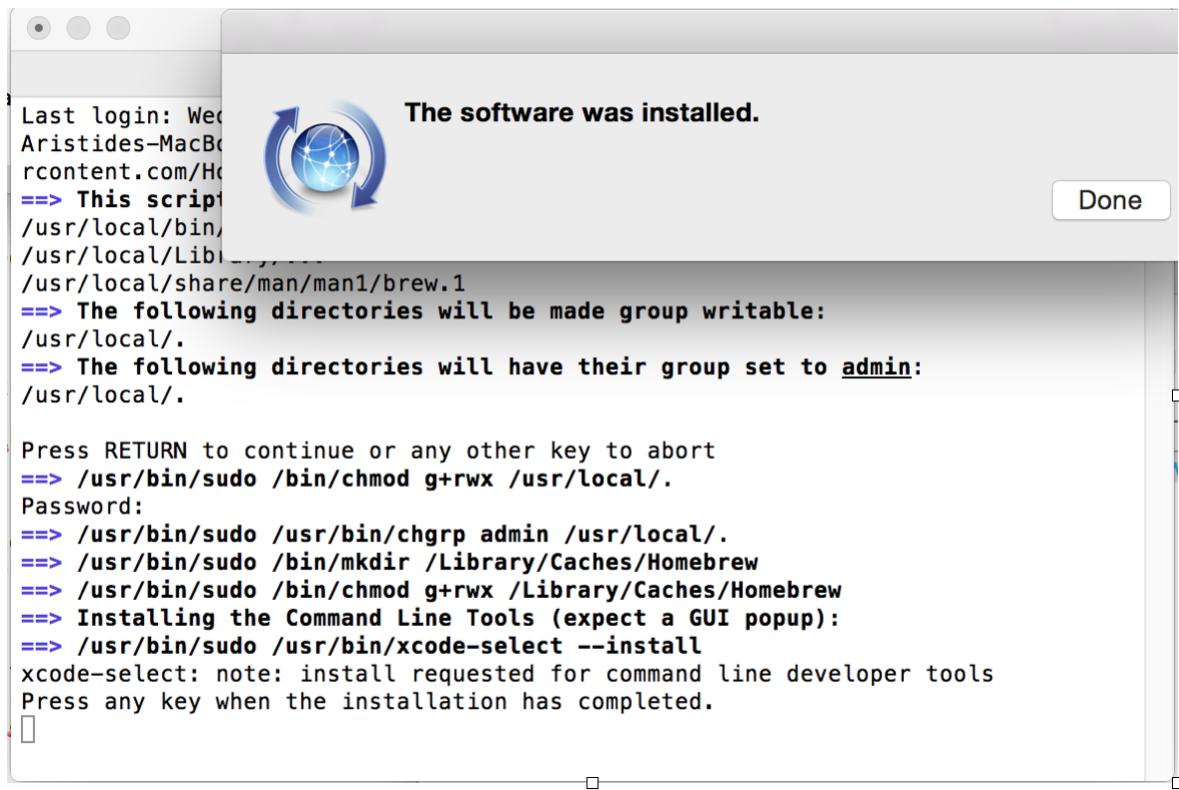
```
Aristides-MacBook-Pro:build yedtoss$ ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
==> This script will install:
/usr/local/bin/brew
/usr/local/Library/...
/usr/local/share/man/man1/brew.1
==> The following directories will be made group writable:
/usr/local/.
==> The following directories will have their group set to admin:
/usr/local/.

Press RETURN to continue or any other key to abort
```

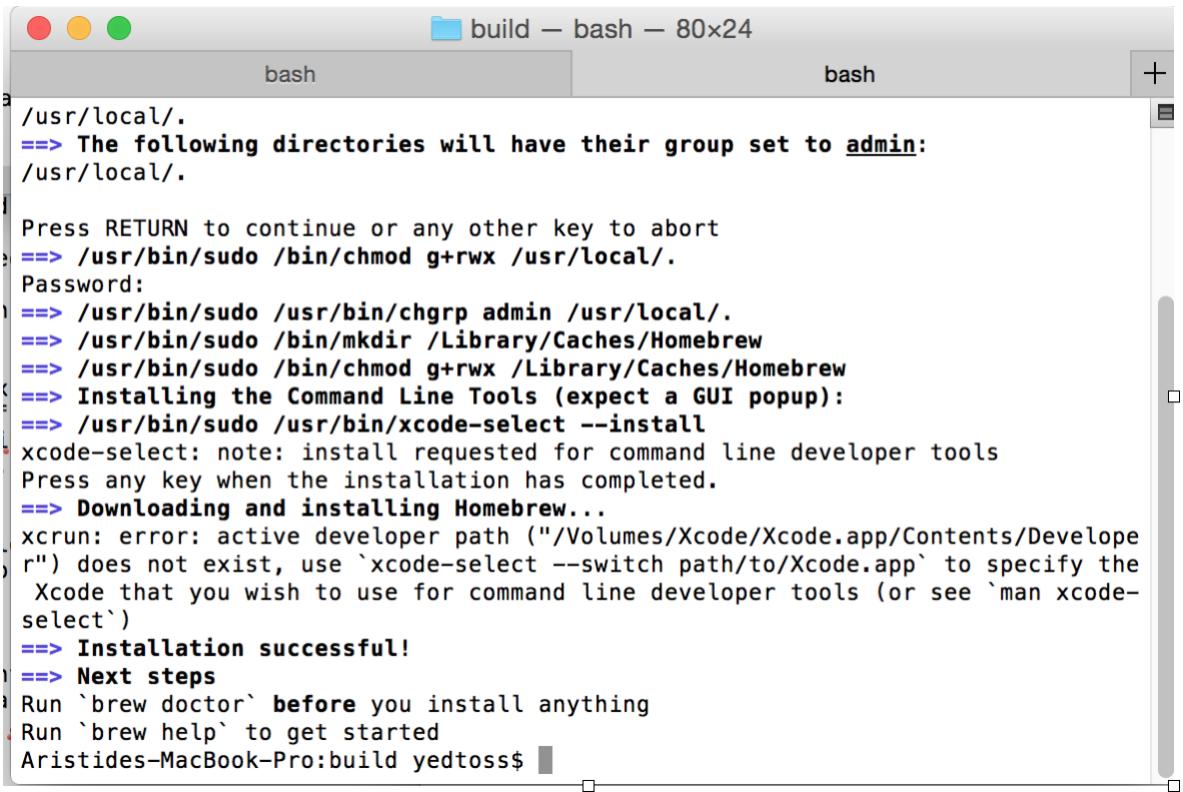
- If you do not have the Xcode command-line tools, you will be prompted to install them as shown below.



- Install the software and make sure the installation is successful as shown below.

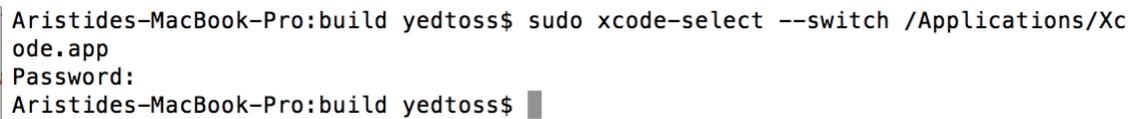


- Press any key. It is possible that you will see an error about the active developer path:



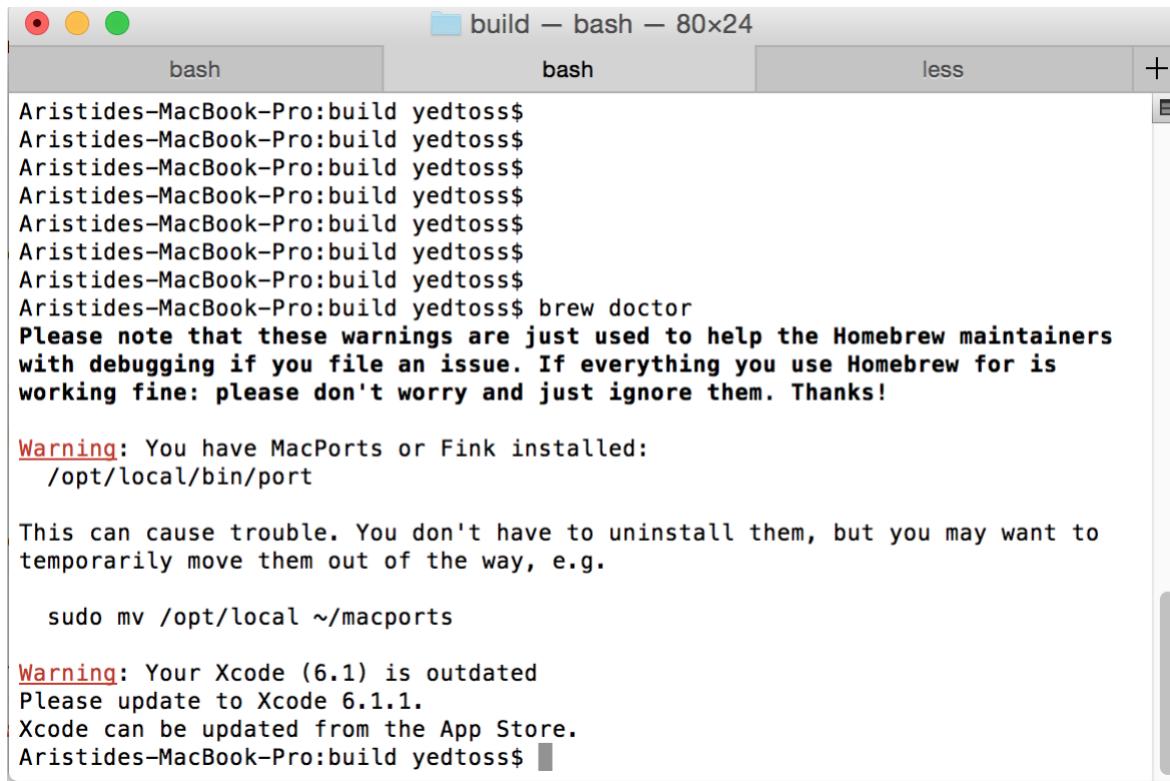
```
/usr/local/.  
==> The following directories will have their group set to admin:  
/usr/local/.  
  
Press RETURN to continue or any other key to abort  
==> /usr/bin/sudo /bin/chmod g+rwx /usr/local/.  
Password:  
==> /usr/bin/sudo /usr/bin/chgrp admin /usr/local/.  
==> /usr/bin/sudo /bin/mkdir /Library/Caches/Homebrew  
==> /usr/bin/sudo /bin/chmod g+rwx /Library/Caches/Homebrew  
==> Installing the Command Line Tools (expect a GUI popup):  
==> /usr/bin/sudo /usr/bin/xcode-select --install  
xcode-select: note: install requested for command line developer tools  
Press any key when the installation has completed.  
==> Downloading and installing Homebrew...  
xcrun: error: active developer path ("~/Volumes/Xcode/Xcode.app/Contents/Developer") does not exist, use `xcode-select --switch path/to/Xcode.app` to specify the Xcode that you wish to use for command line developer tools (or see `man xcode-select`)  
==> Installation successful!  
==> Next steps  
Run `brew doctor` before you install anything  
Run `brew help` to get started  
Aristides-MacBook-Pro:build yedtoss$
```

- If you do get that error, you can correct it by specifying the version of Xcode to use:
[sudo xcode-select –switch /Applications/Xcode.app](#)



```
Aristides-MacBook-Pro:build yedtoss$ sudo xcode-select --switch /Applications/Xcode.app  
Password:  
Aristides-MacBook-Pro:build yedtoss$
```

- Now run this command: [brew doctor](#)



A terminal window titled "build - bash - 80x24" with three tabs: "bash", "bash", and "less". The "bash" tab contains the following text:

```
Aristides-MacBook-Pro:build yedtoss$  
Aristides-MacBook-Pro:build yedtoss$ brew doctor  
Please note that these warnings are just used to help the Homebrew maintainers with debugging if you file an issue. If everything you use Homebrew for is working fine: please don't worry and just ignore them. Thanks!  
  
Warning: You have MacPorts or Fink installed:  
/opt/local/bin/port  
  
This can cause trouble. You don't have to uninstall them, but you may want to temporarily move them out of the way, e.g.  
  
sudo mv /opt/local ~/macports  
  
Warning: Your Xcode (6.1) is outdated  
Please update to Xcode 6.1.1.  
Xcode can be updated from the App Store.  
Aristides-MacBook-Pro:build yedtoss$
```

- Ignore any warnings that may appear. Install GCC and GFortran: [brew install gcc](#). You can ignore warnings about multilib.

```

build — curl — 80x24
bash curl less +
/usr/local/Cellar/gmp/6.0.0a: 15 files, 3.2M
==> Installing gcc dependency: mpfr
==> Downloading https://homebrew.bintray.com/bottles/mpfr-3.1.2-p10.yosemite.bottle.tar.gz
#####
100.0%
==> Pouring mpfr-3.1.2-p10.yosemite.bottle.1.tar.gz
/usr/local/Cellar/mpfr/3.1.2-p10: 24 files, 3.5M
==> Installing gcc dependency: libmpc
==> Downloading https://homebrew.bintray.com/bottles/libmpc-1.0.3.yosemite.bottle.tar.gz
#####
100.0%
==> Pouring libmpc-1.0.3.yosemite.bottle.tar.gz
/usr/local/Cellar/libmpc/1.0.3: 10 files, 380K
==> Installing gcc dependency: isl
==> Downloading https://homebrew.bintray.com/bottles/isl-0.12.2.yosemite.bottle.tar.gz
#####
100.0%
==> Pouring isl-0.12.2.yosemite.bottle.2.tar.gz
/usr/local/Cellar/isl/0.12.2: 55 files, 3.1M
==> Installing gcc dependency: cloog
==> Downloading https://homebrew.bintray.com/bottles/cloog-0.18.1.yosemite.bottle.tar.gz
#####
100.0%
==> Pouring cloog-0.18.1.yosemite.bottle.2.tar.gz
/usr/local/Cellar/cloog/0.18.1: 33 files, 560K
==> Installing gcc
==> Downloading https://homebrew.bintray.com/bottles/gcc-4.9.2_1.yosemite.bottle.tar.gz
#####
61.3%

```

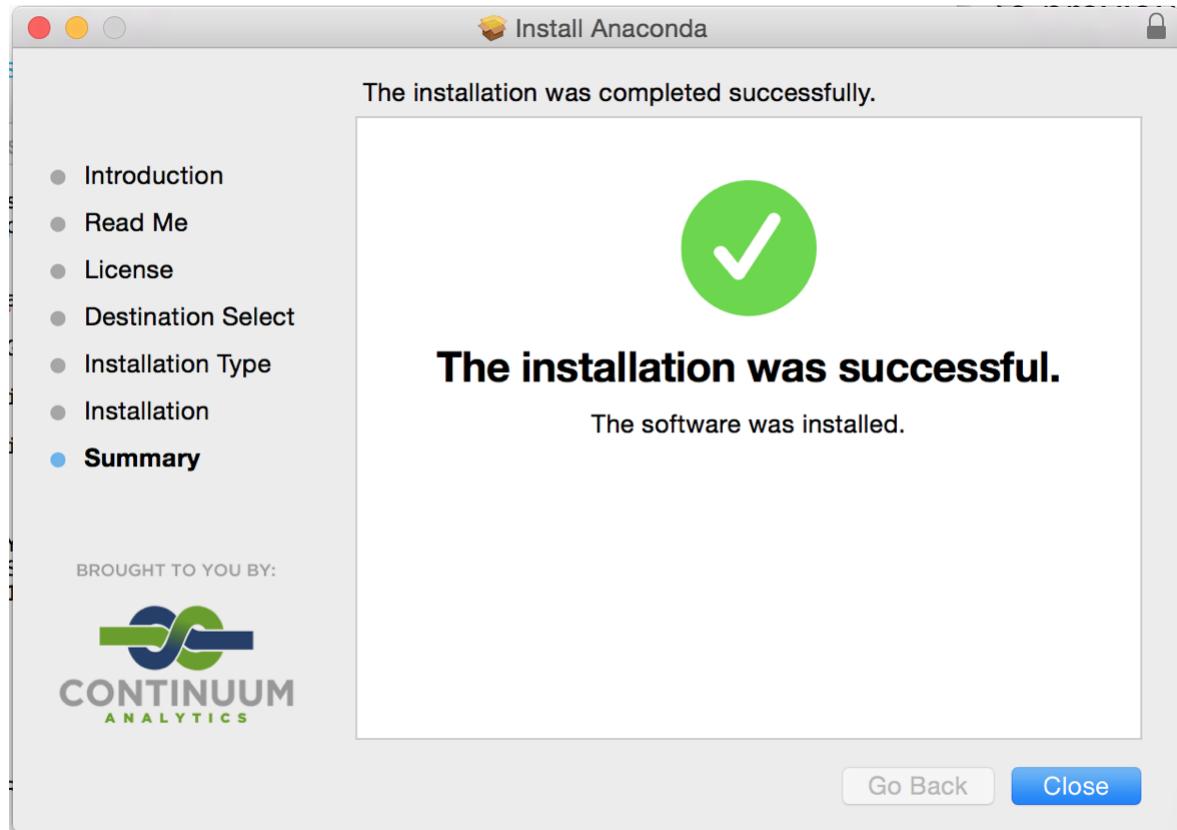
- Install CMake: [brew install cmake](#)

```

build — bash — 80x24
bash bash less +
/usr/local/Cellar/cloog/0.18.1: 33 files, 560K
==> Installing gcc
==> Downloading https://homebrew.bintray.com/bottles/gcc-4.9.2_1.yosemite.bottle.tar.gz
#####
100.0%
==> Pouring gcc-4.9.2_1.yosemite.bottle.tar.gz
==> Caveats
GCC has been built with multilib support. Notably, OpenMP may not work:
  https://gcc.gnu.org/bugzilla/show_bug.cgi?id=60670
If you need OpenMP support you may want to
  brew reinstall gcc --without-multilib
==> Summary
/usr/local/Cellar/gcc/4.9.2_1: 1156 files, 203M
Aristides-MacBook-Pro:build yedtoss$ brew install cmake
==> Installing cmake dependency: xz
==> Downloading https://homebrew.bintray.com/bottles/xz-5.2.0.yosemite.bottle.tar.gz
#####
100.0%
==> Pouring xz-5.2.0.yosemite.bottle.tar.gz
/usr/local/Cellar/xz/5.2.0: 59 files, 1.7M
==> Installing cmake
==> Downloading https://homebrew.bintray.com/bottles/cmake-3.1.2.yosemite.bottle.tar.gz
#####
100.0%
==> Pouring cmake-3.1.2.yosemite.bottle.tar.gz
/usr/local/Cellar/cmake/3.1.2: 1821 files, 30M
Aristides-MacBook-Pro:build yedtoss$ 

```

- Download and install Anaconda for OS X, 64-bit version with Python 2.7 and graphical installer, from <http://continuum.io/downloads>.



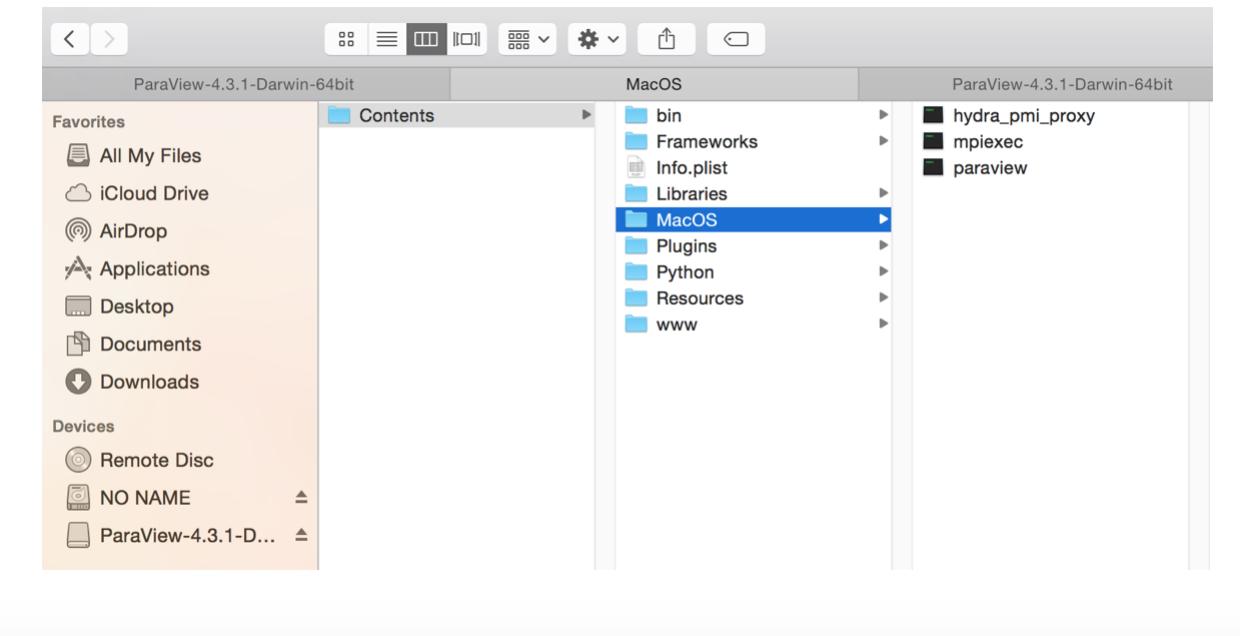
- Now we are going to compile Nemoh Fortran in order to generate the libraries used by the OpenWarp web application. In the following, **\$NEMOH_FORTRAN** is the directory NemohMerged/Nemoh/ within the top-level OpenWarp directory.
- Create a new directory different from **\$NEMOH_FORTRAN**. Let's call this directory **\$FORTRAN_BUILD**.
- Go to **\$FORTRAN_BUILD**: `cd $FORTRAN_BUILD`
- This step is optional for a local testing. However if you want to redistribute or create an installer you MUST do it.

Make sure that the dynamic version of quadmath library is not in your path by running the following command. You may have to adapt 4.9 to the version of gcc installed by brew.

- (One line command) `mv /usr/local/lib/gcc/4.9/libquadmath.0.dylib /usr/local/lib/gcc/4.9/disable_libquadmath.0.dylib`
- (One line command) `mv /usr/local/lib/gcc/4.9/libquadmath.dylib /usr/local/lib/gcc/4.9/disable_libquadmath.dylib`
- Run this command to compile Nemoh Fortran: `cmake -DCMAKE_Fortran_COMPILER="gfortran" $NEMOH_FORTRAN` If you get a warning about CMake policy, ignore it. Generate the Nemoh Fortran library by running: `make` This causes `libnemoh.dylib` to be created.

```
[ 67%] Building Fortran object CMakeFiles/nemoh.dir/Solver/Core/Gaussm3.f90.o
[ 70%] Building Fortran object CMakeFiles/nemoh.dir/Solver/Core/GREEN_FUNCTION.f90.o
[ 72%] Building Fortran object CMakeFiles/nemoh.dir/Solver/Core/SOLVE_BEM_HIGH.f90.o
[ 75%] Building Fortran object CMakeFiles/nemoh.dir/Solver/Core/ODE.f90.o
[ 78%] Building Fortran object CMakeFiles/nemoh.dir/Solver/Core/COMPUTE_INFLUENCE_ODE.f90.o
[ 81%] Building Fortran object CMakeFiles/nemoh.dir/Solver/Core/SOLVE_BEM_INFIDIRECT.f90.o
[ 83%] Building Fortran object CMakeFiles/nemoh.dir/Solver/Core/SOLVE_BEM_ITERATIVE.f90.o
[ 86%] Building Fortran object CMakeFiles/nemoh.dir/Solver/Core/SOLVE_BEM.f90.o
[ 89%] Building Fortran object CMakeFiles/nemoh.dir/Solver/NEMOH.f90.o
[ 91%] Building Fortran object CMakeFiles/nemoh.dir/commonFiles/Environment.f90.o
[ 94%] Building Fortran object CMakeFiles/nemoh.dir/commonFiles/Results.f90.o
[ 97%] Building Fortran object CMakeFiles/nemoh.dir/Solver/Core/ALLOCATE_DATA.f90.o
[100%] Building Fortran object CMakeFiles/nemoh.dir/Solver/Core/DEALLOCATE_DATA.f90.o
Linking Fortran shared library libnemoh.dylib
[100%] Built target nemoh
Aristides-MacBook-Pro:build yedtoss$
```

- Copy `libnemoh.dylib` from `$FORTRAN_BUILD` to the `lib/` directory inside the Anaconda installation root: `cp $FORTRAN_BUILD/libnemoh.dylib /Users/bob/anaconda/lib` (Replace "bob" with your own user name.)
- Download the binary installer of ParaView version ≥ 4.1 for OS X from <http://www.paraview.org/download/> and install it. Next, copy the `paraview.app/` folder from the ParaView directory to `$ROOT/src/bundled/` so that we can invoke ParaView to do visualization. (You may need to use sudo if you are performing the copy operation on the command line.) The files under `$ROOT/src/bundled/paraview.app/` should be organized as below:



- We have to start the OpenWarp server before using the web application. To start the server: Make sure you are using Anaconda's version of Python by running: `export PATH=/Users/bob/anaconda/bin:$PATH` (Replace "bob" with your own user name.) Now run: `python -version` You should see Anaconda in the output as shown here:
- Run the following command to verify that the library path is correctly set up:
`(test -e /Users/bob/anaconda/lib/libnemoh.dylib && echo 'Success') || echo 'Error: Nemoh library not found'`. (As before, replace "bob" with your own user name.)

```
Aristides-MacBook-Pro:src yedtoss$ 
Aristides-MacBook-Pro:src yedtoss$ export LD_LIBRARY_PATH=/Users/yedtoss/anaconda/lib
Aristides-MacBook-Pro:src yedtoss$ export LDFLAGS="-L/Users/yedtoss/anaconda/lib"
Aristides-MacBook-Pro:src yedtoss$ echo $LD_LIBRARY_PATH
/Users/yedtoss/anaconda/lib
Aristides-MacBook-Pro:src yedtoss$ echo $LDFLAGS
-L/Users/yedtoss/anaconda/lib
Aristides-MacBook-Pro:src yedtoss$ (test -e /Users/yedtoss/anaconda/lib/libnemoh.dylib && echo 'Nemoh is successfully found' ) || echo 'Error Nemoh Library not found'
Nemoh is successfully found
Aristides-MacBook-Pro:src yedtoss$
```

If you get an error message here, it means you did not correctly set up the path to the Nemoh library.

- Make sure the nglbmesh binary is runnable: `chmod +x $ROOT/src/bundled/mesh-generator/build/nglbmesh` (Recall that `$ROOT` is the top-level directory of the OpenWarp package.)

- To prevent dylib errors, go to the directory `$ROOT/src/bundled/mesh-generator/lib/` and run:

```
python update_dylib.py
```

```
Aristides-MacBook-Pro:openwarpgui yedtoss$ chmod +x src/bundled/mesh-generator/b
uild/nglib-mesh
Aristides-MacBook-Pro:openwarpgui yedtoss$ cd src/bundled/mesh-generator/lib
Aristides-MacBook-Pro:lib yedtoss$ python update_dylib.py
Aristides-MacBook-Pro:lib yedtoss$
```

- Install the remaining Python dependencies by going to `$ROOT/src/` and running: `pip install -r requirements.txt` This step is required before running the OpenWarp server for the first time. You can skip it when running the server on subsequent occasions.

```
Aristides-MacBook-Pro:src yedtoss$ pip install -r requirements.txt
  Downloading/unpacking CherryPy (from -r requirements.txt (line 1))
    Downloading CherryPy-3.6.0.tar.gz (432kB): 432kB downloaded
      Running setup.py (path:/private/var/folders/81/0r8qm66x40d88d523jj7t6c80000gn/
T/pip_build_yedtoss/CherryPy/setup.py) egg_info for package CherryPy

Requirement already satisfied (use --upgrade to upgrade): numpy in /Users/yedtoss/anaconda/lib/python2.7/site-packages (from -r requirements.txt (line 2))
Requirement already satisfied (use --upgrade to upgrade): cython in /Users/yedtoss/anaconda/lib/python2.7/site-packages (from -r requirements.txt (line 3))
Requirement already satisfied (use --upgrade to upgrade): h5py in /Users/yedtoss/anaconda/lib/python2.7/site-packages (from -r requirements.txt (line 4))
Installing collected packages: CherryPy
  Running setup.py install for CherryPy
    changing mode of build/scripts-2.7/cherryd from 644 to 755

    changing mode of /Users/yedtoss/anaconda/bin/cherryd to 755
Successfully installed CherryPy
Cleaning up...
Aristides-MacBook-Pro:src yedtoss$
```

- To start the server, run: `sudo python main.py` If you get an error, make sure that port 80 is not being used by another process.

```

----- -----
Aristides-MacBook-Pro:src yedtoss$ sudo python main.py
Password:
running build_ext
[25/Feb/2015:11:56:04] ENGINE Listening for SIGHUP.
[25/Feb/2015:11:56:04] ENGINE Listening for SIGTERM.
[25/Feb/2015:11:56:04] ENGINE Listening for SIGUSR1.
[25/Feb/2015:11:56:04] ENGINE Bus STARTING
[25/Feb/2015:11:56:04] ENGINE Started monitor thread 'Autoreloader'.
[25/Feb/2015:11:56:04] ENGINE Started monitor thread '_TimeoutMonitor'.
[25/Feb/2015:11:56:04] ENGINE Serving on http://127.0.0.1
[25/Feb/2015:11:56:04] ENGINE Bus STARTED
127.0.0.1 -- [25/Feb/2015:11:56:06] "GET /index.html HTTP/1.1" 200 2396 "" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10) AppleWebKit/600.1.25 (KHTML, like Gecko) Version/8.0 Safari/600.1.25"
127.0.0.1 -- [25/Feb/2015:11:56:07] "GET /js/scripts.js HTTP/1.1" 200 22812 "http://127.0.0.1/index.html" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10) AppleWebKit/600.1.25 (KHTML, like Gecko) Version/8.0 Safari/600.1.25"
127.0.0.1 -- [25/Feb/2015:11:56:07] "GET /js/bootstrap.js HTTP/1.1" 200 60681 "http://127.0.0.1/index.html" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10) AppleWebKit/600.1.25 (KHTML, like Gecko) Version/8.0 Safari/600.1.25"
127.0.0.1 -- [25/Feb/2015:11:56:07] "GET /css/bootstrap.css HTTP/1.1" 200 13873 "http://127.0.0.1/index.html" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10) AppleWebKit/600.1.25 (KHTML, like Gecko) Version/8.0 Safari/600.1.25"

```

If you get an error message reading "library not found for -lnemoh" or "No module named solver_fortran", make sure you have correctly exported the path to Nemoh in the current terminal session as explained earlier.

5.3 Linux deployment from scratch

It is enough to run the script install.sh inside **\$ROOT** while being connected to the Internet. Note that you are supposed to use that script from the pre-built package of Linux as explained in ??; otherwise you should download and install Paraview yourself before running the script. Only tested on Ubuntu 14.04 64 bits.

6 Using Install Script

Install Script is used to download, extract and install all the prerequisite software using one script only. It also build native libnemoh and solver_fortran according to operating system type.

6.1 Windows

Installer.bat is the install script that is certified for Windows 7,10 server 2008 and 2012 x64 machines. Please make sure you run CMD as an administrator:

- Click Start, click All Programs, and then click Accessories
- Right-click Command prompt, and then click Run as administrator.

- If the User Account Control dialog box appears, confirm that the action or displays is what you want, and then click Continue.

Installer.bat brings-down all the prerequisites to get started and configure path and dlls to get started. installer.bat could be edited to add/modify new logics.

- 1. Determine whether 32-bit or 64-bit OS
- 2. Sets 7zip and curl in the Path from local repo
- 3. Download and install Anaconda
- 4. Download, Extracts and installs Mingw
- 5. Download, Extracts and install
- 6. Create gFortran workspace
- 7. Build libnemoh.dll and libnemoh.dll.a
- 8. Copies libnemoh.dll inside Anaconda and MinGW installation directory
- 9. Copies 19 dependent libraries inside Anaconda/DLLs folder
- 10. Install essential python libraries from requirement.txt
- 11. Download and install Paraview
- 12. Builds solver_fortran.pyd
- 13. Ready to OpenWarp.

After the installation is done, Execute run.bat from the source directory to Open GUI. Similarly, to run automated testscripts, you can run testscript_1.bat and testscript_2.bat. To know more about CLI, refer OpenWARPCLIdocumentation.pdf

6.2 OS X

installer.sh is the install script that is certified in OS X 10.10. It downloads and configures all the prerequisites in following sequence. Installer.sh could be edited to add/modify new logic/url etc in the installation steps/

- 1. Checks if the OS type is Darwin
- 2. Installs command line tools for Xcode
- 3. Install HomeBrew
- 4. Install Curl
- 5. Install Gcc, Gfortran and Cmake

- 6. Installs Anaconda commandline installer
- 7. Build libnemoh.dylib
- 8. Download and install ParaView
- 9. Install python libraries from requirements.txt
- 10. Configure native path to libnemoh for solver_fortran.so
- 11. Builds solver_fortran
- 12. Ready to run OpenWarp.

After the installation is done, Execute run.sh from the source directory to Open GUI. Similarly, to run automated testscripts, you can run testscript_1.sh and testscript_2.sh. To know more about CLI, refer OpenWARPCLIdocumentation.pdf

6.3 Linux

installer.sh is the install script for Linux too. It helps in downloading and installing all the prerequisites in following sequence

- 1. Install Gcc and GFortran
- 2. Install Cmake
- 3. Install Blas and Lapack
- 4. Install nglib
- 5. Install OpenCascade Community Edition
- 6. Install HDF5-tools
- 7. Install VTK
- 8. Install Python Dependencies
- 9. Install mesh-generator nglib-mesh
- 10. Build native libnemoh.so
- 11. Export LD_Library_Path
- 12. Build solver_fortran.so
- 13. Ready to run OpenWarp

After the installation is done, Execute run.bat from the source directory to Open GUI. Similarly, to run automated testscripts, you can run testscript_1.sh and testscript_2.sh. To know more about CLI, refer OpenWARPCLIdocumentation.pdf

7 Deployment from existing package

Instead of building from scratch, you can first install the old installer using section ??

Then follow these steps:

7.1 Windows

7.1.1 Recompiling NemohFortran and NemohPython

- You need to make sure to replace all new codes in **\$INSTALL_DIR**/OpenWarp by the one in the submission (the new one.)
- Follow instructions of section 5.1 to build a local copy. Replace **\$NEMOH_FORTRAN** by **\$INSTALL_DIR**/OpenWarp/NemohImproved/Nemoh; **\$ROOT/src** should be replaced by **\$INSTALL_DIR**/OpenWarp/openwargui. The python version of nemoh should be **\$INSTALL_DIR**/OpenWarp/openwargui/nemoh. Anaconda should be **\$INSTALL_DIR**/OpenWarp/Anaconda
- Now copy the newly generated libnemoh.dll and libnemoh.dll.a to **\$INSTALL_DIR**/OpenWarp/Anaconda/DLLs.
- If your code needs any additional DLLS make sure to put them there

7.1.2 Repackaging the installer

- Copy the directory src/Windows/installer to **\$INSTALL_DIR** such that **\$INSTALL_DIR**/OpenWarp and **\$INSTALL_DIR**/installer exists.
- Download and Install Inno Setup version equal or greater than 5.5.5 from <http://www.jrsoftware.org/isdl.php>
- Open **\$INSTALL_DIR**/installer/openwarp.iss with Inno Setup (You may double-click on the file)
- Inno Setup will open. Click on Build -> Compile (menu). Once done the installer will be in **\$INSTALL_DIR**/OpenWarp/installer/Output/OpenWarp.exe

7.2 OS X

7.2.1 Recompiling NemohFortran and NemohPython

- You need to make sure to replace all new codes in **\$INSTALL_DIR**/OpenWarp by the one in the submission (the new one.)

- Follow instructions of section 5.2 to build a local copy. Replace **\$NEMOH_FORTTRAN** by **\$INSTALL_DIR**/OpenWarp/NemohImproved/Nemoh; **\$ROOT/src** should be replaced by **\$INSTALL_DIR**/OpenWarp/openwargui. The python version of nemoh should be **\$INSTALL_DIR**/OpenWarp/openwargui/nemoh. Anaconda should be **\$INSTALL_DIR**/OpenWarp/Anaconda
- Now copy the newly generated libnemoh.dylib to **\$INSTALL_DIR**/OpenWarp/anaconda/lib
- Then **VERY IMPORTANTLY** run in a terminal `install_name_tool -change "libnemoh.dylib" "../anaconda/lib/libnemoh.dylib"` from **\$INSTALL_DIR**/OpenWarp/openwargui/nemoh/solver_fortran.so from **\$INSTALL_DIR**/OpenWarp/openwargui/nemoh

7.2.2 Packaging the installer

- Copy the directory src/MAC/installer to **\$INSTALL_DIR** such that **\$INSTALL_DIR**/OpenWarp and **\$INSTALLDIR/installer** exists.
- Download and install Packages (version $\geq 1.1.1$) from <http://www.macupdate.com/app/mac/34613/packages>
- Open **\$INSTALL_DIR**/OpenWarp/installer/OpenWarp with Packages (You can double-click on it)
- In the menu click on Build \rightarrow Build or Type Command +B
- The installer will start building and once done the MAC OS X packages will be in **\$INSTALL_DIR**/OpenWarp/installer/build/OpenWarp.pkg

7.3 Linux

Here the file installer.sh is taking care of everything. In most cases you don't have to do anything extras in case the code is updated. In any case just make sure installer.sh and run.sh are accurate and can be used to install and run the new code in a clean system.

7.3.1 Recompiling NemohFortran and NemohPython

Not needed as the installer.sh take care of that automatically.

7.3.2 Packaging the installer

You just need to compress the directory **\$INSTALL_DIR**.

7.4 General Guide about the installer build/packaging procedure

OpenWarp is composed of three main parts: the Fortran part of Nemoh whose source is available in `$INSTALL_DIR/OpenWarp/NemohImproved/Nemoh`; the Python part of Nemoh whose source is available in `$INSTALL_DIR/OpenWarp/openwarpgui/nemoh` and the OpenWarp GUI part whose source is available in `$INSTALL_DIR/OpenWarp/openwarpgui/openwarp`. If you do not update the Fortran part of Nemoh or the file `$INSTALL_DIR/OpenWarp/openwarpgui/nemoh/solver_fortran.pyx` then the only step you have to do to rebuild the installer is to package it directly. There is no need to do any re-compilation steps. When you update the Fortran or `$INSTALL_DIR/OpenWarp/openwarpgui/nemoh/solver_fortran.pyx` you need to recompile them and copy the generated library in a well-defined location. See the relevant OS guide for more information.

In case you added some python dependencies not currently available, you would need to package them in the installer. For Linux you can update the relevant `$INSTALL_DIR/OpenWarp/openwarpgui/requirements.txt` and it will be installed automatically by `$INSTALL_DIR/OpenWarp/installer.sh`

For Windows and MAC OS X you would need to install those dependencies in the provided Anaconda site-packages available in `$INSTALL_DIR/OpenWarp/Anaconda` and `$INSTALL_DIR/OpenWarp/anaconda` respectively. If you add some external libraries as dependencies you need to make sure they are available to end users. In Windows you can just copy them to `$INSTALL_DIR/OpenWarp/Anaconda/DLLs`.

In MAC OS X you can just copy them to `$INSTALL_DIR/OpenWarp/anaconda/lib`. Make sure there are static or update their dependencies using `install_name_tool` (You can see an example in the MAC OS X specific instructions). For Linux/Ubuntu, it is preferred to use `apt-get install` to make those external libraries available to end users.

8 Configuration

The three parts of the application can all be configured using the web application GUI. In this section, we describe the configurable parameters for each part.

8.1 Meshing parameters

Input File : This is the input file geometry for which we want to generate quadrilateral elements. The accepted input file formats are IGS, STEP, and STL.

Output File : Enter the name of the output file without any file extension. This should be a bare file name without a directory path preceding it.

Fineness : The target fineness of the mesh.

MaxH : The maximum height of the quadrilateral elements in the mesh.

MinH : The minimum height of the quadrilateral elements in the mesh.

Grading : The grading of the mesh.

Use Tolerance ?: Whether or not to use tolerance when generating the mesh. When the absolute value of the difference between two values is less than the tolerance, we consider them to be equal.

Tolerance : Volume tolerance expressed as a number from 0 to 1.

8.2 Simulation parameters

Rho : The fluid's specific volume in KG/m^3 .

G : Gravity in m/s^2 .

Depth : The depth of the water in meters. Enter 0 to indicate infinite depth.

XEFF, YEFF : The x and y coordinates, in meters, of the point where the wave is measured.

Name of Mesh File : The name of the mesh file generated in the meshing step.

Number of Points and Number of Panels : The number of points in the mesh and the total number of panel or quadrilaterals in the mesh. These values are automatically filled in.

Surge: The freedom of translation along the X axis (moving forward and backward).

Sway : The freedom of translation along the Y axis (moving left and right).

Heave : The freedom of translation along the Z axis (moving up and down).

Roll About A Point : The freedom of rotation along the X axis (pivoting side to side).

Pitch About A Point : The freedom of rotation along the Y axis (tilting forward and backward).

Yaw About A Point : The freedom of rotation along the Z axis (swiveling left and right).

Force In X, Y, Z Direction : Three-dimensional coordinates of force.

Moment Force In X, Y, Z Direction About A Point : Three-dimensional coordinates of moment force.

Number of Lines of Additional Information Currently not supported. Should be set to 0.

Number of Wave Frequencies, MIN, MAX : Respectively the number of wave frequencies, the minimum value of each wave frequency, and the maximum value of each wave frequency.

Number of Wave Directions, MIN, MAX : Respectively the number of wave directions, the minimum value of each wave direction, and the maximum value of each wave direction.

Indiq_solver : The method to use for solving linear equation. Use 0 to indicate the Gauss Method, 1 to indicate the GMRES method, and 2 to indicate the GMRES method with FMM acceleration. Option 2 is not yet supported.

TOL_GMRES : The value of the tolerance to use for the GMRES method. The tolerance is used to determine convergence. IRES : Restart parameter for GMRES. MAXIT: Maximum iterations for GMRES.

Sav_potential : Use 0 or 1 to indicate whether or not the potential should be saved in the output.

GREEN_TABULATION_NUMX : Represents the number of points in the X direction in the tabulated data.

GREEN_TABULATION_NUMZ : Represents the number of points in the Z direction in the tabulated data.

GREEN_TABULATION_SIMPSON_NPOINTS : Represents the number of sub-intervals used to approximate the Green's function integral using Simpson's rule.

USE_ODE_INFLUENCE_COEFFICIENTS Indicate whether or not to use the ODE method to compute the influence coefficients.

USE_HIGHER_ORDER : Whether or not to use the higher-order panel method.

NUM_PANEL_HIGHER_ORDER : The number of panels per patch in the higher-order method.

B_SPLINE_ORDER : The order of the B-spline for the potential in the higher-order panel method.

USE_DIPOLES_IMPLEMENTATION : Whether or not to use the dipole implementation.

THIN_PANELS : A list containing the indices of panels which are thin dipoles. Indices are zero-based. Set the list to [-1] to indicate that all panels are thin dipoles.

COMPUTE_DRIFT_FORCES : Whether or not to compute the drift forces.

COMPUTE_YAW_MOMENT : Whether or not to compute the yaw moment.

8.3 Postprocessing parameters

IRF : Whether or not to compute the IRF, i.e., the infinite frequency added mass and the impulse response function for the radiation force.

TIME Step : The time step used for the IRF computation.

Duration : The duration used for the IRF computation.

Show Pressure : Whether or not to output the pressure forces.

Kochin Function : The number of angles to use in computing the Kochin function. Set 0 to disable this computation.

Min Angle, Max Angle : The minimum and maximum angle to use in computing the Kochin function.

Number of Points In X, Y Direction : Number of points in each direction for the free-surface visualization. Set 0 to disable the computation.

Dimensions of Domain In X, Y Direction : The domain dimension in each direction, in degrees.

8.4 Other and Logging parameters

There are only two parameters currently for the Configuration screen and they all support logging. We have:

Logging level : Pick "DEBUG" or "INFO" to set the logging level to the respective value

Clear old logs : Whether or not to delete the old log files

The logging level can also be configured using the configuration file **\$NEMOH_Python/logging.json**

When run from the web interface the logs are located by default in the user home in OpenWarpFiles/logs

When run from the command line the logs are located in the configured base test directory in **\$NEMOH_Python/settings.py**

9 Usage

9.1 Start the application

Make sure the server has been started: `python main.py` This command must be executed from the `$ROOT/source/openwarpgui` directory Open a web browser and go to: `http://127.0.0.1/index.html` You will see the following.

- To launch the application on Windows, go to where you have installed it (by default C:/Program Files (x86)/OpenWarp) and double-click "run.bat"

You can then access the GUI using the url `http://127.0.0.1:8386/index.html`

Don't use Internet Explorer as it is known not to work with the GUI.

- To launch the application on Linux, go to where you have installed it (The directory where you extracted the archive) from a terminal and run "bash run.sh" You can then access the GUI using the url `http://127.0.0.1:8386/index.html`
- To launch the application on MAC OS X, go to where you have installed it (by default /Applications/OpenWarp) and double-click on "run" file You can then access the GUI using the url `http://127.0.0.1:8386/index.html`



You can quit the application by clicking the orange Quit button. Quitting results in:



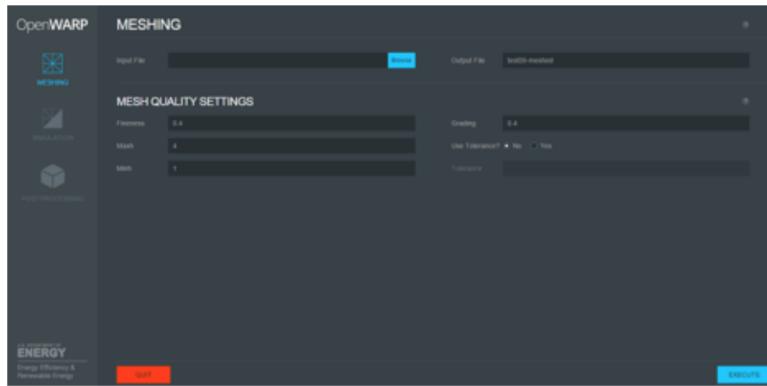
You must restart the Python server before using the application again.

9.2 Start meshing

Click the Meshing button:



This is the meshing page:



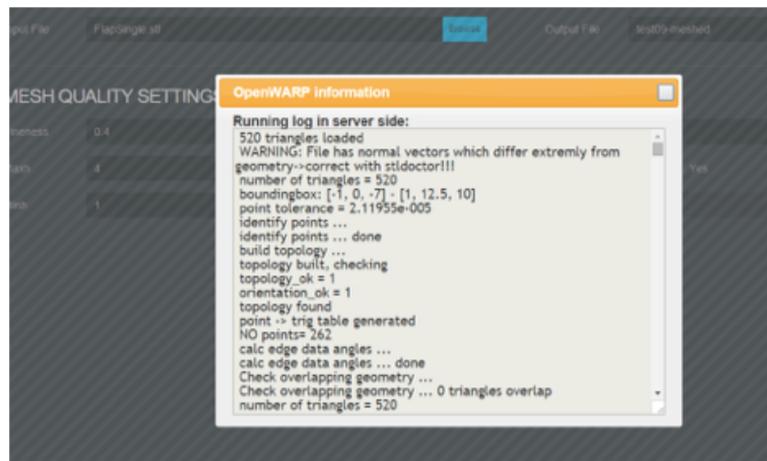
Select the input file by clicking Browse and selecting a file. Set the other parameters according to your needs.

9.3 Execute meshing

Click the blue Execute button.

9.4 View the meshing results

Once meshing is complete, you will see a popup displaying a log:



The meshing converts the input file to all formats recognised by the application. It also converts the input file to a mesh containing only quadrilateral elements without any triangle. The output directory is **\$ROOT/src/user_data/meshing_TIMESTAMP_HASH** where **\$ROOT** is the top-level directory of the OpenWarp package. The timestamp is in the format YYYYMMDDHHMM. Navigate to the directory with the latest timestamp. Inside the directory, you will find:

- Two files ending in .vti, containing the mesh in VTP format. The "quad" version contains only quadrilateral elements.
- Two files ending in .vtk, containing the mesh in VTK format.

- Two files ending in .dat, the format recognized by Nemoh.
- Two files ending in .gdf, containing the mesh in GDF format.
- Two files ending in .stl, containing the mesh in stl format.

Each file is named OUTPUT_FILE_NAME.EXTENSION or OUTPUT_FILE_NAME-quad.EXTENSION, where OUTPUT_FILE_NAME is the output file name configured before meshing and EXTENSION corresponds to the file format. For each pair of files, the "quad" version contains only quadrilateral elements.

9.5 Start the simulation

Click the Simulation button:



This is the resulting page:

LOAD CASES TO BE SOLVED

Roll About A Point: 0 0 -7.5
Pitch About A Point: 0 0 -7.5
Yaw About A Point: 0 0 -7.5
Number of Lines of Additional Information: 0

Number of Wave Frequencies: 2
Number of Wave Directions: 1
Min: 0.1 radians
Max: 2 radians
Min: 0 degrees
Max: 0 degrees

CALCULATION PARAMETERS

Indq_solver: 0
TOL_GMRES: 5e-07
Sse_potential: 1
GREEN_TABULATION_NUMX: 328
GREEN_TABULATION_NUMZ: 46
IRES: 20
MANT: 100
USE_DOF_INFLUENCE_COEFFICIENTS: 0
GREEN_TABULATION_NUMX: 328
GREEN_TABULATION_NUMZ: 46
NUM_PANEL_HIGHER_ORDER: 1
USE_Dipoles_IMPLEMENTATION: 0
COMPUTE_DRIFT_FORCES: 0

EXECUTE

9.6 Execute the simulation

Click the Execute button at the bottom right of the page.

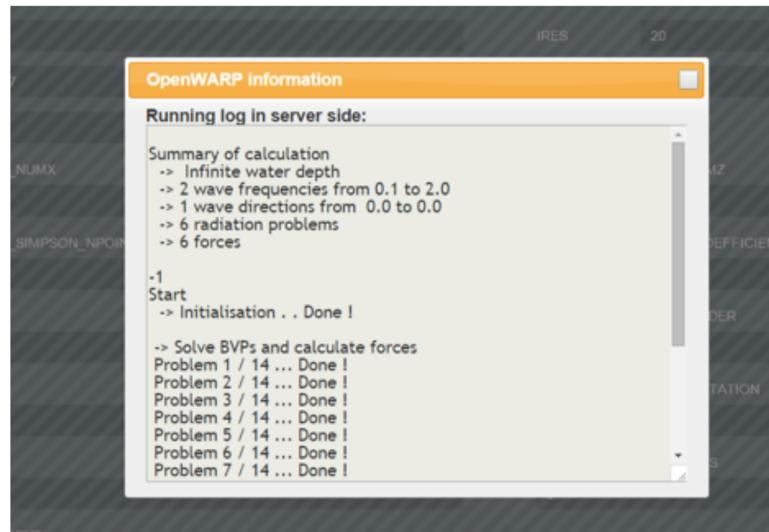
CALCULATION PARAMETERS

Indq_solver: 0
TOL_GMRES: 5e-07
Sse_potential: 1
GREEN_TABULATION_NUMX: 328
GREEN_TABULATION_NUMZ: 46
IRES: 20
MANT: 100
USE_DOF_INFLUENCE_COEFFICIENTS: 0
NUM_PANEL_HIGHER_ORDER: 1
USE_Dipoles_IMPLEMENTATION: 0
COMPUTE_DRIFT_FORCES: 0

EXECUTE

9.7 View the simulation results

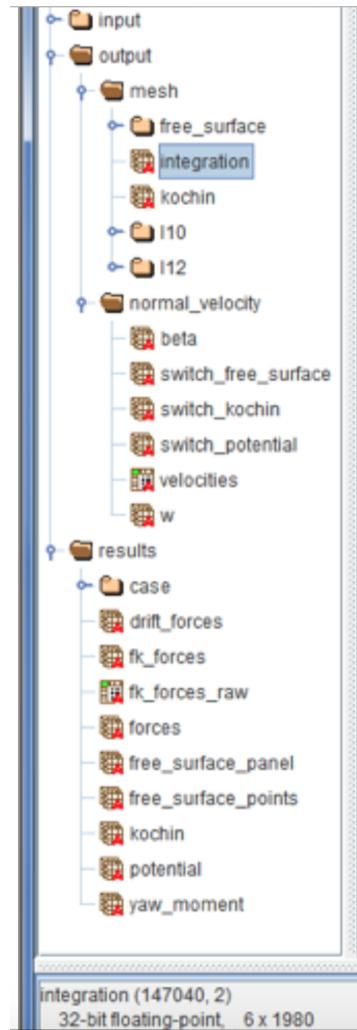
Once the simulation has completed, you will see a popup containing the calculation log:



The output directory is `$ROOT/src/user_data/simulation_TIMESTAMP_HASH` where `$ROOT` is the top-level directory of the OpenWarp package. The timestamp is in the format `YYYYMMDDHHMM`. Navigate to the directory with the latest timestamp. The main result at this step is the `db.hdf5` file inside that directory. You can open it with HDFView:



You will see that it contains three main groups: the input group, the output group containing intermediate results generated by the application, and the results group containing final results. In each group, you can see datasets with their description as shown in the figure below.

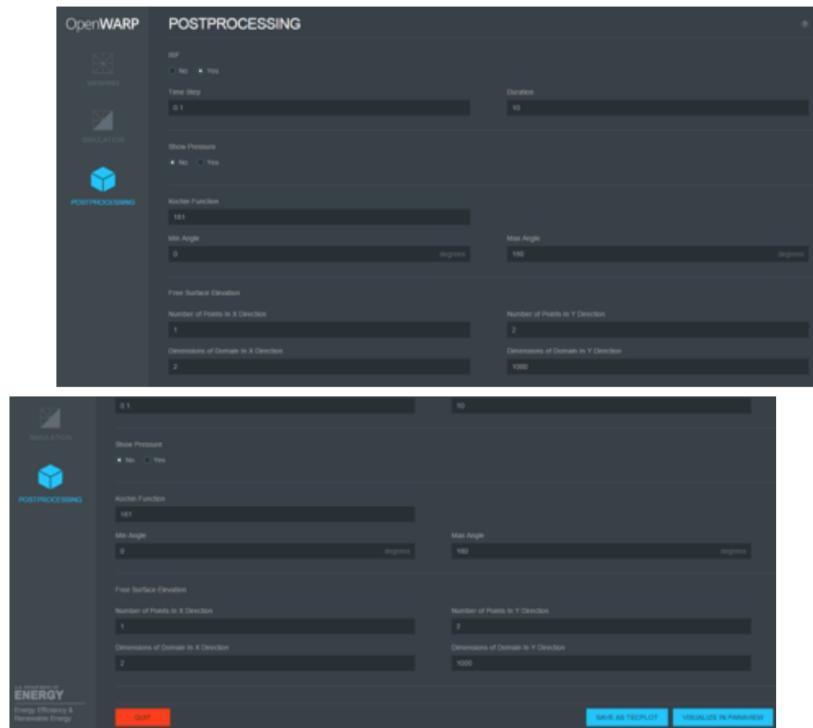


9.8 Postprocessing

Click the Postprocessing button:

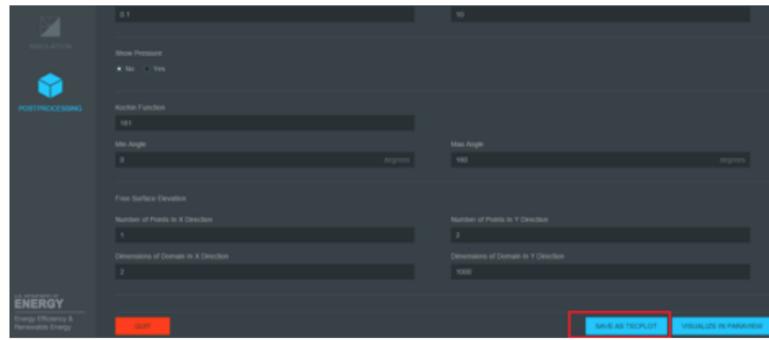


You should see the following:



9.9 Generate TEC outputs

Generate the TEC file by clicking on "SAVE AS TECPLOT". Note that before generating a TEC file, you must have executed a simulation in the current session.



9.10 Postprocessing results

Once the postprocessing is done, you can manually examine the results in the output directory `$ROOT/src/user_data/simulation_TIMESTAMP_HASH` where `$ROOT` is the top-level directory of the OpenWarp package. The timestamp is in the format YYYYMMDD-HHMM. Navigate to the directory with the latest timestamp. You will find the following output files:

results/irf.tec : This file contains the infinite frequency added mass and the impulse response function for the radiation force.

results/WaveField.tec : The wave field in TECPLOT format.

results/diffractionforce.tec : The diffraction force for the diffraction problems.

results/excitationforce.tec : The excitation force for the diffraction problems.

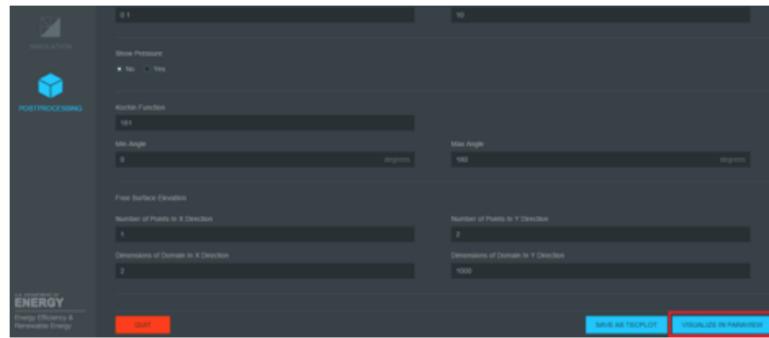
results/radiationcoefficients.tec : The added mass and damping forces for the radiation problems.

results/fkforce.tec : The Froude-Krylov forces for the diffraction problems.

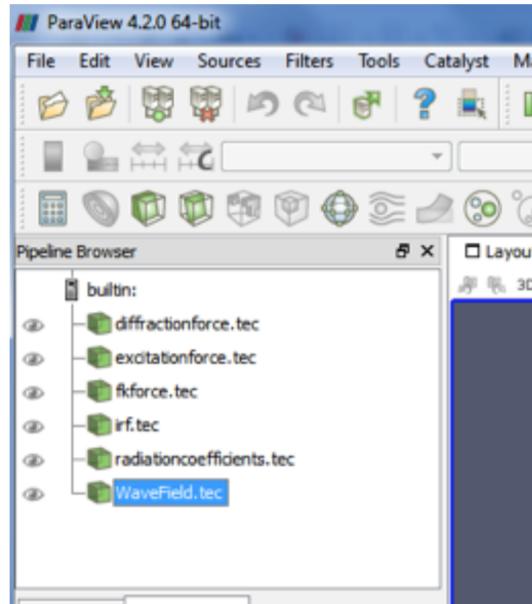
mesh/mesh.tec : The mesh file in TECPLOT format. It contains tables of nodes and connections.

9.11 Visualize the generated TEC files

Instead of manually viewing the postprocessing output files, you can visualize them with ParaView. After generating the TEC files, click on "VISUALIZE IN PARAVIEW":

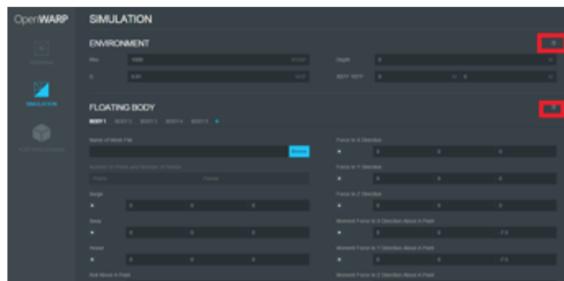


ParaView will start and you will find the resulting forces on the left side:



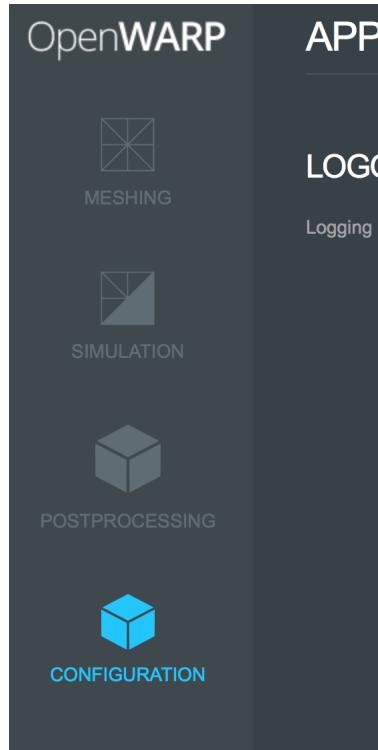
9.12 Getting help

You can click on the Help icons displayed on each page:

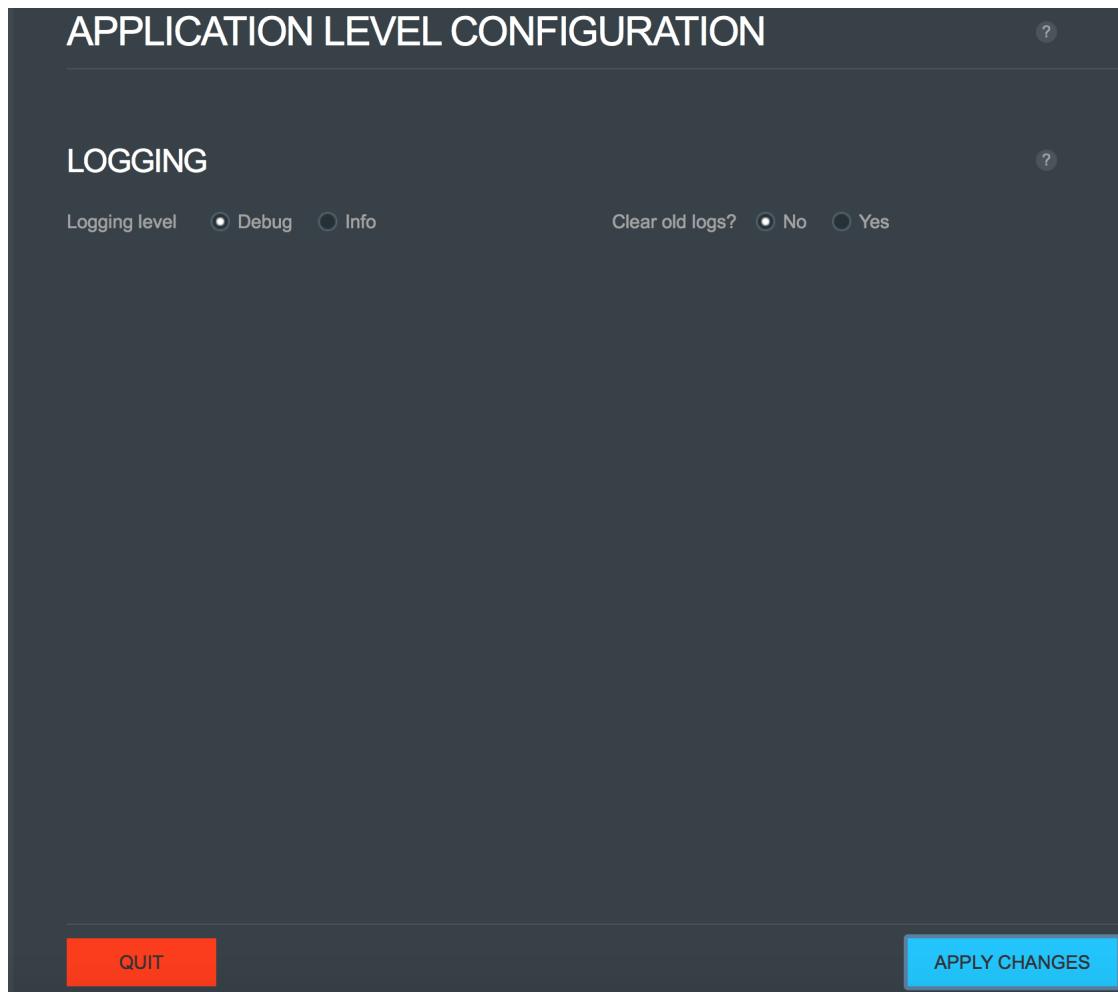


9.13 Logging Support

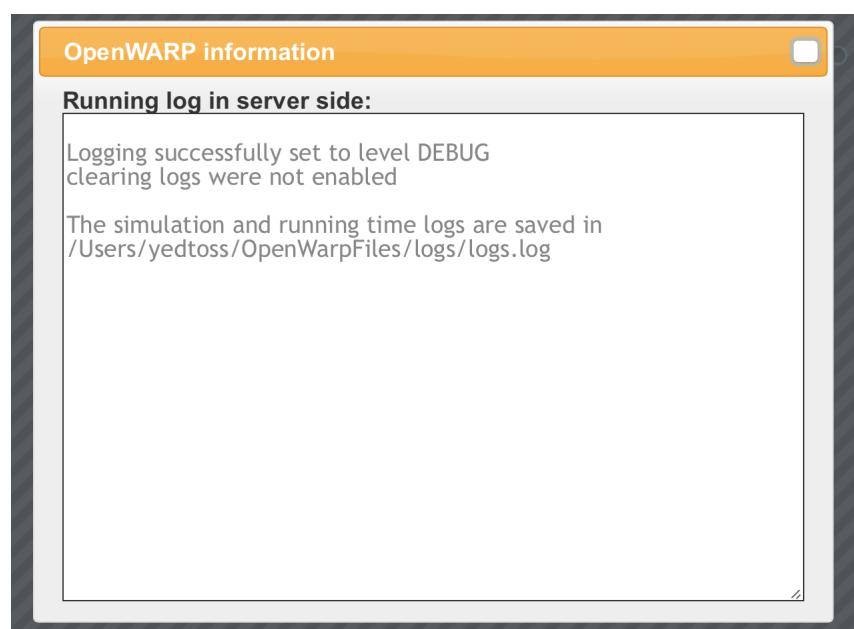
To configure the logging click on the CONFIGURATION screen as shown below:



After setting the configuration as your wish, you should click on "APPLY CHANGES" to make sure your changes are taken into account.



Once done, you will get a status of the change request as well as the location where the logs are saved.



10 Python CLI

In order to run the application through the command line instead of through the GUI, the application also provides a command line interface that is suitable to be used when running batch jobs.

You can find this in the source/openwarpgui/ directory. The filename is openwarp_cli.py

Before being able to run the CLI, you should setup your environment library 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.

10.1 CLI Verification

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. Following have been executed in the OSX environment but it should be possible to replicate in others too.

1. Basic Testing Run python openwarp_cli.py test_files/configs/cylinder.json This test is just to make sure that all the configurations are working correctly

2. 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.
3. 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
4. same_conf_diff_geometry.json This is to test that we can run multiple different simulations with the same configurations on different geometry.
5. flapsingle_igs.json This is to test the flapsingle geometry. This will take an awfully lot of time (More than 10h) to complete and require a lot of memory.
6. Error Message When the configuration is invalid or the command misused an appropriate error message is shown

11 Automated TestScript

Under source/automated test folder, there are 11 test cases implemented. Refer theory documentation for details on each Test.

11.1 Repository

- source/installer.sh - Install all the pre-requisite software required
- source/testscript.sh - Script to trigger all the test from test1 to test6
- source/testscript2.sh - Script to trigger all the test from test7 to test11
- source/automated test/ - Directory contains workspace for each tests. All the output results gets created here.
- source/automated_test.log - log file for testscript
- source/automated_test2.log - log file for testscrpt2
- doc/deployment_guide.pdf - Deployment guide to start OpenWarp in general.

11.2 Dependencies

All the prerequisite software and there installations are listed above under dependencies. Following modules are dependent to tests:

- 1. openwarp_cli.py - All test, except test1 uses openwarp_cli.py. For details informations, refer Python CLI section in this doc or under Contest_Output/PythonCLI. Testscript.sh source run.sh to set the environment variables and dependencies. Testscript2.sh also set environment variables to successfully run openwarp_cli.py
- 2. Bemio - Test are written in such a way that, it has merged only the subpart of Bemio here and there according to individual test requirement.
- 3. H5DIFF- H5DIFF utility comes with HDF5-TOOLS set. To install in ubuntu ,sudo apt-get install hdf5-tools command can be used. although it covered in installer.sh
- 4. Wamit testruns - Wamit testruns directory to make use of Wamit's test and its output to compare with OpenWarp output.

Following sections briefly describe each test case:

11.3 Test 1: Converting GDF to OpenWarp Format

11.3.1 Objective of Test case

To convert Wamit' GDF file to DAT file to make it compatible with OpenWarp.

11.3.2 Steps to execute the test

Run testscript.sh

11.3.3 Output

All the subsequent GDF files of Wamit are convert to .DAT format under automated_test/testruns/openwarp/**.dat Outputlog: automated_test.log

11.4 Test 2: Wamit Test Input and Output

11.4.1 Objective of Test case

To convert Wamit's output in H5 format. To run Wamit's GDF files(by convert to DAT from test 1) To compare Wamit's Ouput to new OpenWarp output

11.4.2 Steps to execute the test

Run testscript.sh, it will automatically run Test2 after executing test1 cases.

11.4.3 Output and Comparison

.H5 files under testruns/db.hdf5 db.hdf5 files under automated_test/testruns/openwarp/openwarpH5 test*.log under Test2 (it contains comparison against all 11 types of outputs). The comparison is done objects by objects using H5DIFF tool. Outputlog: automated_test.log

11.5 Test 3: Testing HigherOrder Implementation in OpenWarp

11.5.1 Objective of Test case

To test difference in output for B_SPLINE Order under higher order implementation.

11.5.2 Steps to execute the test

Run testscript.sh, it will automatically run Test3 to create an output by using Wamit's test11 to test25

11.5.3 Output and Comparison

testlogs under test3.log Outputlog: automated_test.log

11.6 Test 4: Testing the Dipoles/Thin Panels implementation in OpenWarp

11.6.1 Objective of Test case

To test differences in output while dipoles and thin panels implementation

11.6.2 Steps to execute the test

Run testscript.sh

11.6.3 Output and Comparison

testlogs under test4.log Outputlog: automated_test.log

11.7 Test 5: Testing Irregular Frequency Removal in OpenWarp

11.7.1 Objective of Test case

To test difference in values of objects with irregular frequency in higher as well as lower order implementation

11.7.2 Steps to execute the test

Run testscript.sh

11.7.3 Output and Comparison

testlogs under test5.log Outputlog: automated_test.log

11.8 Test 6: Testing Mean Drift Forces and Yaw moment Implementation in OpenWarp

11.8.1 Objective of Test case

To test difference in values in horizontal drift forces and mean yaw moment by momentum integration.

11.8.2 Steps to execute the test

Run testscript.sh

11.8.3 Output and Comparison

testlogs under automated_Test/Test6 Outputlog: automated_test.log

11.9 Test 7: Computation of the Green Function using ordinary differential equation in OpenWarp

11.9.1 Objective of Test case

To determine the difference in values of Green Function when ODE method is used to calculate influence coefficients.

11.9.2 Steps to execute the test

Run testscript2.sh

11.9.3 Output and Comparison

testlogs under automated_test/Test7 Outputlog: automated_test2.log

11.10 Test 8: Testing Green Function Tabulation Implementation

11.10.1 Objective of Test case

To see the difference in tabulated data makes under Green function

11.10.2 Steps to execute the test

Run testscript2.sh

11.10.3 Output and Comparison

testlogs under automated_test/Test8 Outputlog: automated_test2.log

11.11 Test 9: 2Bodies

11.11.1 Objective of Test case

To execute simulation and postprocessing for common values and multiple simulation case.

11.11.2 Steps to execute the test

Run testscript2.sh. automated_test/Test9/2bodies.json has the updated values from Nemoh.cal

11.11.3 Output and Comparison

testlogs under automated_test/Test9 Outputlog: automated_test2.log

11.12 Test 10: Cylinder

11.12.1 Objective of Test case

To execute simulation and postprocessing for cylindrical geometry.

11.12.2 Steps to execute the test

Run testscript2.sh, automated_test/Test10/cylinder.json has the updated values from Nemoh.cal

11.12.3 Output and Comparison

testlogs under automated_test/Test10 Outputlog: automated_test2.log

11.13 Test 11: NonSymmetrical

11.13.1 Objective of Test case

To execute simulation and postprocessing for NonSymmetrical geomerty

11.13.2 Steps to execute the test

Run testscript2.sh, automated_test/Test11/nonsymmetrical.json has the updated values from Nemoh.cal

11.13.3 Output and Comparison

testlogs under automated_test/Test11 Outputlog: automated_test2.log

12 Steps to update OW from Nemoh 2014 to Nemoh 2016

When OW was created, we used Nemoh version 2014. At the time of writing this document, the version of Nemoh is 2016. Thus, the following outlines the steps to follow when including the updates carried out in Nemoh 2016 into OpenWarp.

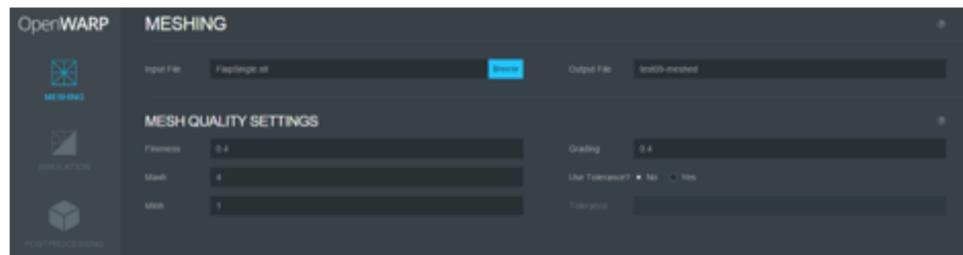
1. First, look for the new code implemented in Nemoh using mercurial diff tool
2. Next, check if they were implemented in the python package or the Fortran part of OW. Preprocessor and postprocessor code in Nemoh are in python part of OW whereas most of the Solver code is in Fortran part of OW. After this step, one should have identified the file and lines where it is implemented / needs to be implemented in OW.
3. Check if the fixed / new code is relevant to OW. Most of the Solver related code in Nemoh is relevant to OW. Preprocessor and postprocessor code are not always relevant.
4. We also need to keep in mind the new features implemented in OW. For example, OW Fortran solver does not write / save to files and as such code changes should be ignored in such aspects. Also, keep in mind Fortran code in OW used Fortran 2003 and uses standard functions instead of vendor specific functions as much as possible. As a result, before changing a function, make sure that OW was not using an equivalent standard function. OW also replaced some functions with their recommended 2003 equivalent. Watch out for these and ignore them. Finally, OW avoids the use of GOTO statement and such changes can also be ignored.
5. As a general rule of thumb, the only fixes to add are the ones that change the computation, not the ones that are doing the same computation using a different approach / style of coding.
6. After all the assessment has been carried out and the changes to carry out have been conclusively determined, apply the changes.

13 Practical Example

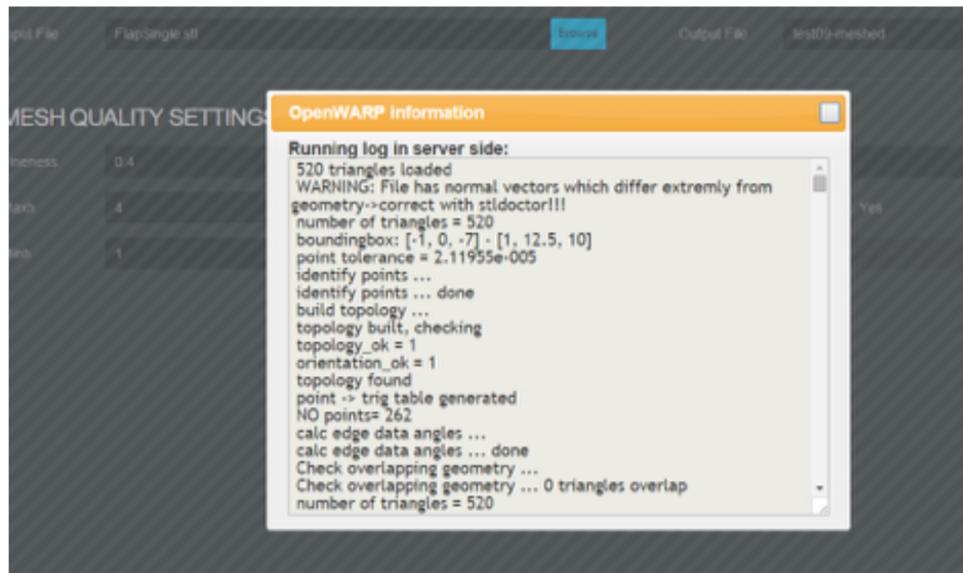
We are going to use the FlapSingle example available in `$ROOT/test_files` where `$ROOT` is the top-level directory of the OpenWarp package. It is also available in `$INSTALL_DIR/OpenWarp/test_files`. First, we will generate the mesh of the body. Next, we will run the simulation. Finally, we will generate the TEC files and visualize them.

13.1 Mesh Generation

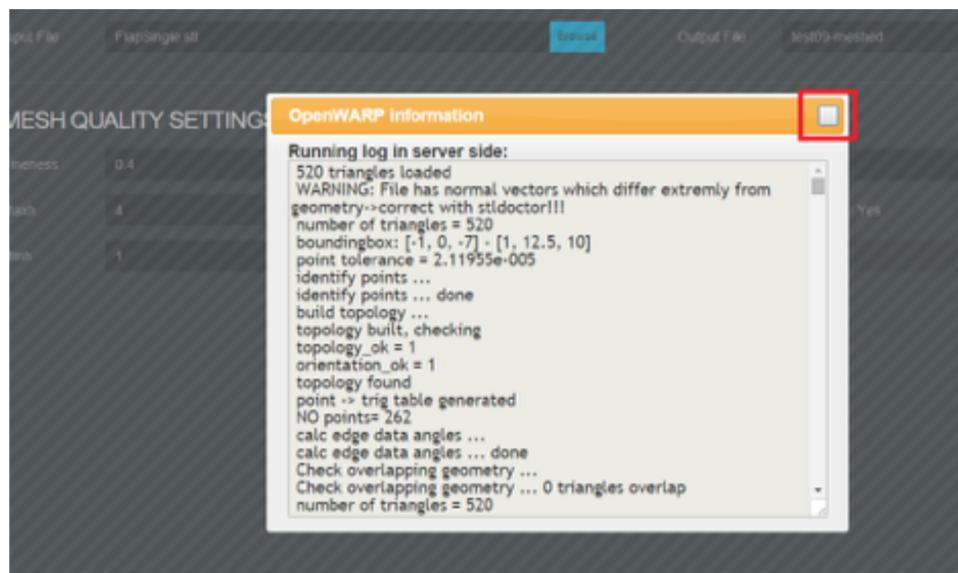
Go to the Meshing page, click on Browse, and choose `$ROOT/test_files/mesh/FlapSingle.stl` where `$ROOT` is the top-level directory of the OpenWarp package and Execute the meshing:



You will see the results shown like this :

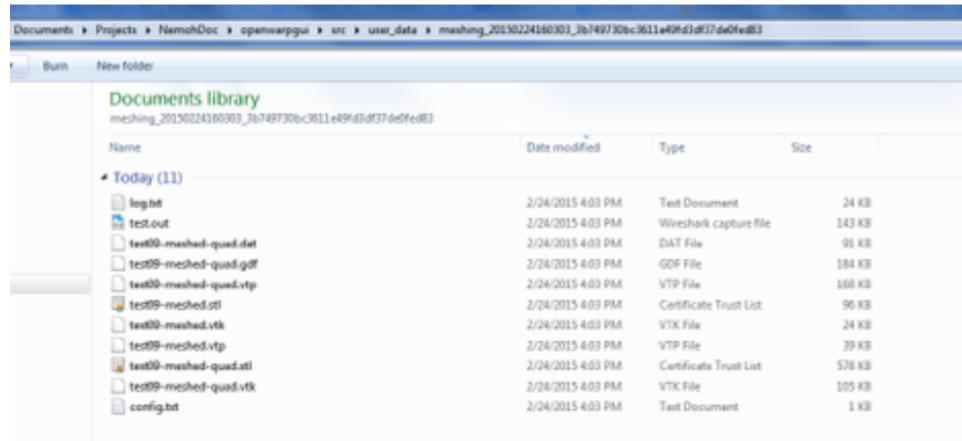


Click the button in the corner of the popup to close it:

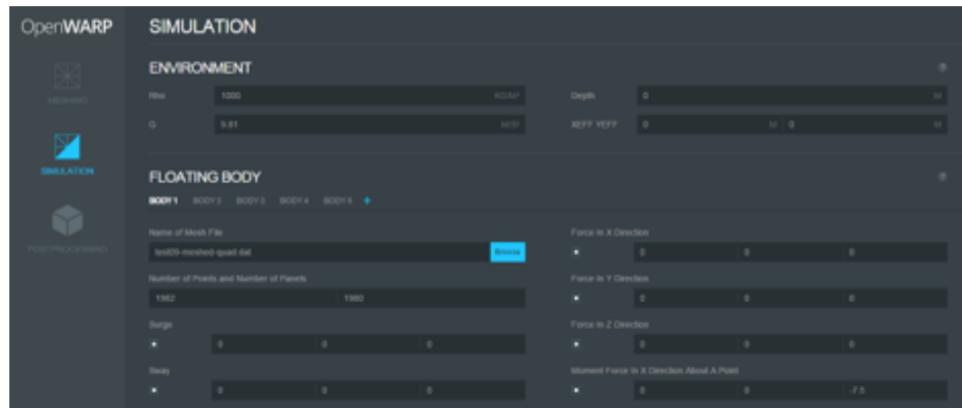


13.2 Run the simulation

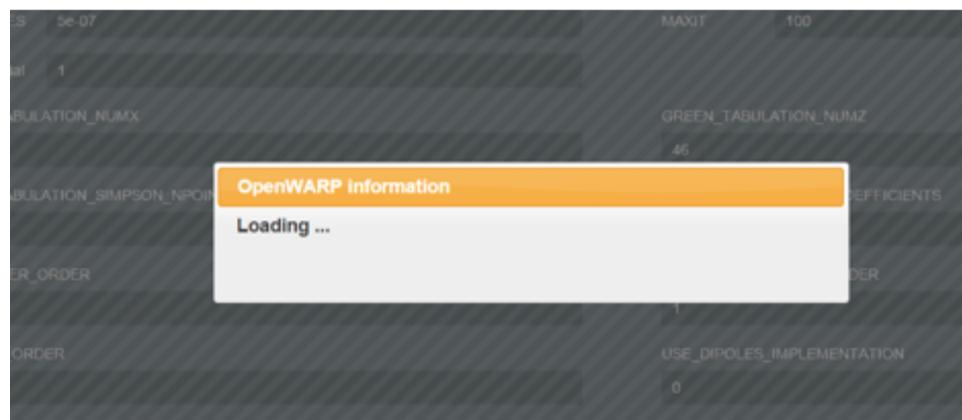
Go to the simulation page and click on Browse to load the generated mesh file. The generated mesh file is inside the \$HOME/OpenWarpFiles/user_data/meshing_TIMESTAMP_HASH where \$HOME is your home directory. Go to the directory with the latest timestamp.



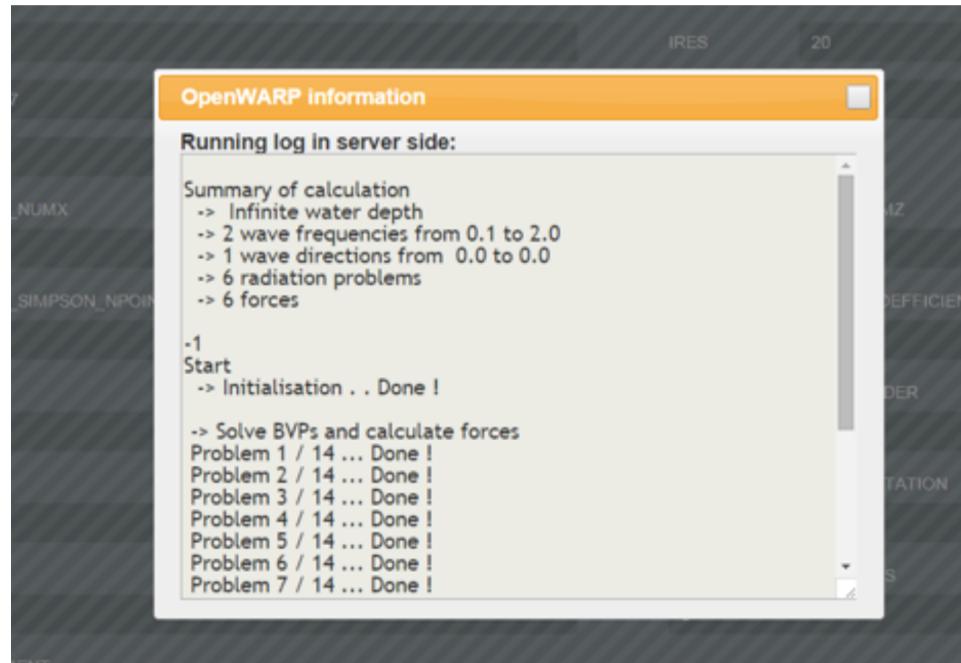
Look for a .dat file. It is named flap-meshed-quad.dat in our case.



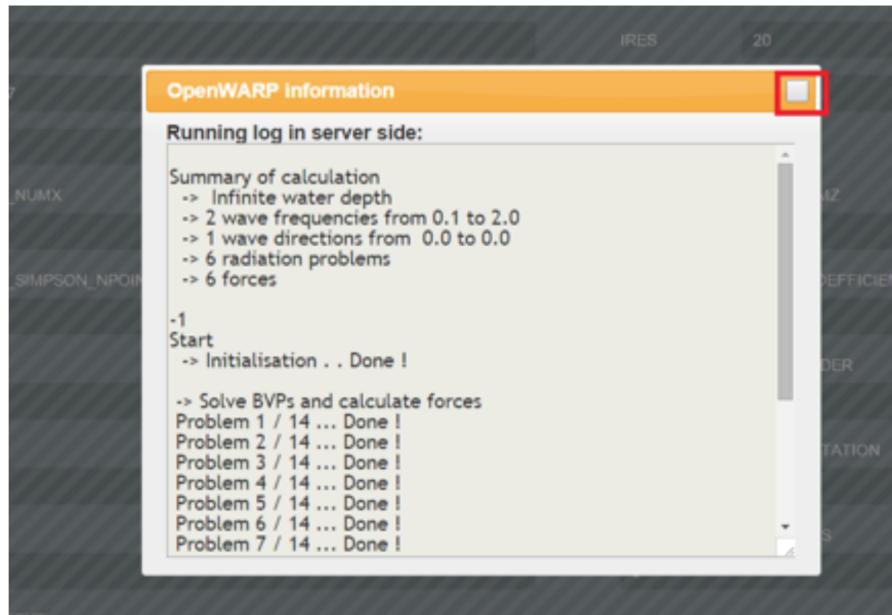
Run the simulation by clicking on "EXECUTE". You should see this:



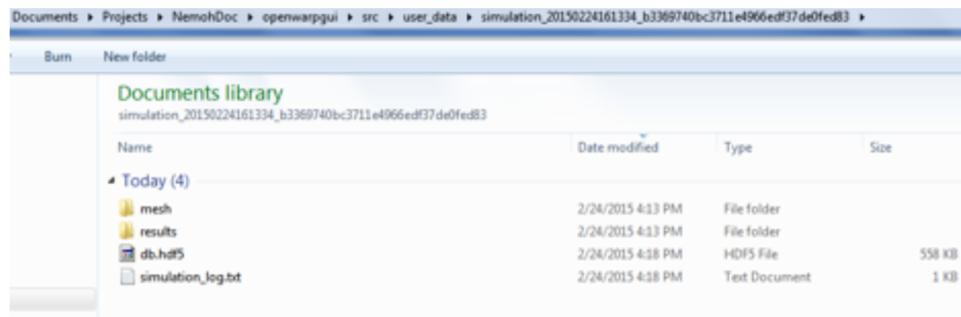
When the simulation is done (You should be patient as it will take a few minutes), the results are summarized in a popup:



Close the popup:

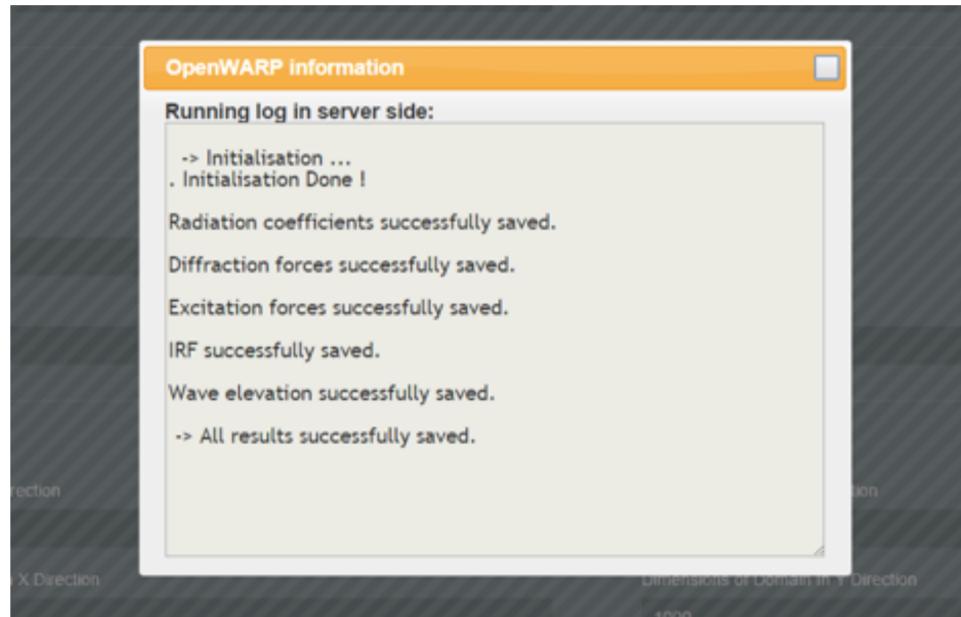


Inside the directory `$HOME/OpenWarpFiles/user_data/simulation_TIMESTAMP_HASH`, you will see the generated simulation results where `$HOME` is the home directory of the current user. Find `db.hdf5` inside the directory with the latest timestamp; it contains the results.

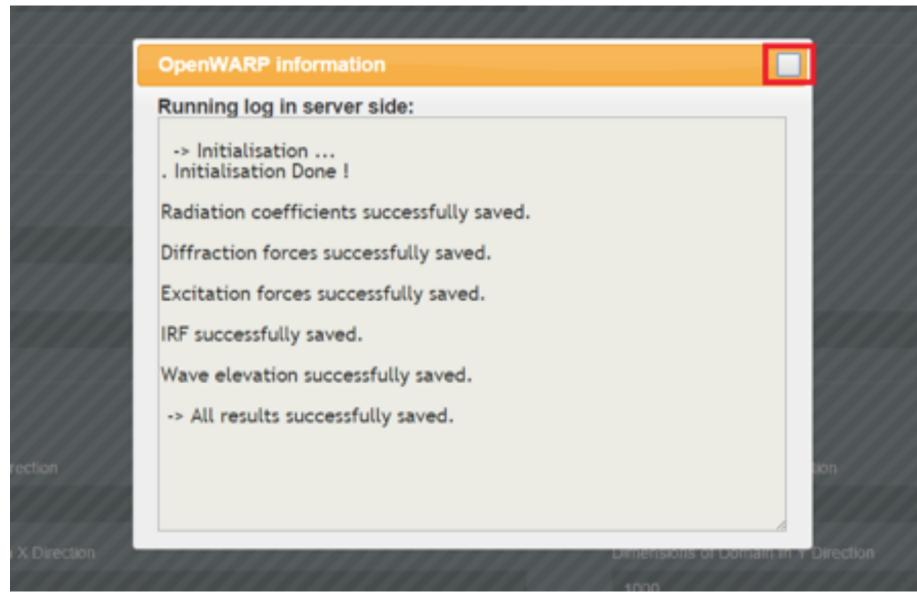


13.3 Generate TEC files and run postprocessing

Open the postprocessing page. Leave the default values. Generate the TEC files by clicking "Save as TECPLOT". When postprocessing is done, you get a popup:

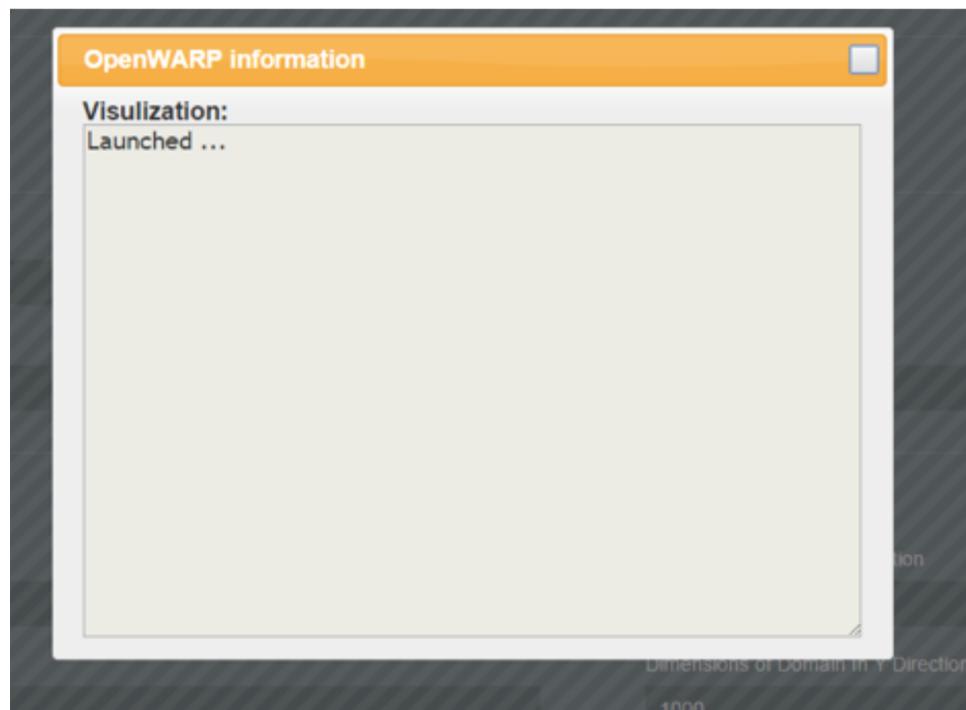


Close it.



13.4 Visualize the generated tec files

Click on "VISUALIZE IN PARAVIEW". You will see this while ParaView opens:



Once ParaView is open, you will see the generated forces on the left:

