Contents

[1- INTRODUCTION](#h.30j0zll)

[2- OVERVIEW](#h.1fob9te)

[3- NOTATIONS](#h.3znysh7)

[4- DEPENDENCIES](#h.2et92p0)

[Windows](#h.tyjcwt)

[OS X](#h.3dy6vkm)

[5. WINDOWS DEPLOYMENT](#h.4d34og8)

[6. OS X DEPLOYMENT](#h.2s8eyo1)

[7. CONFIGURATION](#h.17dp8vu)

[7.1. Meshing parameters](#h.3rdcrjn)

[7.2. Simulation parameters](#h.26in1rg)

[7.3. Postprocessing parameters](#h.lnxbz9)

[8. USAGE](#h.35nkun2)

[8.1. Start the application](#h.1ksv4uv)

[8.2. Start meshing](#h.44sinio)

[8.3. Execute meshing](#h.2jxsxqh)

[8.4. View the meshing results](#h.z337ya)

[8.5. Start the simulation](#h.3j2qqm3)

[8.6. Execute the simulation](#h.1y810tw)

[8.7. View the simulation results](#h.4i7ojhp)

[8.8. Postprocessing](#h.2xcytpi)

[8.9. Generate TEC outputs](#h.1ci93xb)

[8.10. Postprocessing results](#h.3whwml4)

[8.11. Visualize the generated TEC files](#h.2bn6wsx)

[8.12. Getting help](#h.qsh70q)

[9. EXAMPLE](#h.3as4poj)

[9.1. Mesh Generation](#h.1pxezwc)

[9.2. Run the simulation](#h.49x2ik5)

[9.3. Generate TEC files and run postprocessing](#h.2p2csry)

[9.4. Visualize the generated tec files](#h.147n2zr)

# 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.

# 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.

# NOTATIONS

The following notations are used in this document.

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

**$NEMOH\_FORTRAN**: the directory **$ROOT**/NemohMerged/Nemoh/

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

**$MINGW\_ROOT**: the directory where MinGW will be installed.

# 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.

## Windows

* MinGW 4.8.1 [http://sourceforge.net/projects/MinGWbuilds/files/host-windows/releases/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/>

## OS X

* 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 >=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 installed by brew
* Anaconda (with Python 2.7) >= 2.1.0 <http://continuum.io/downloads>
* ParaView >=4.1 <http://www.paraview.org/download/>

# WINDOWS DEPLOYMENT

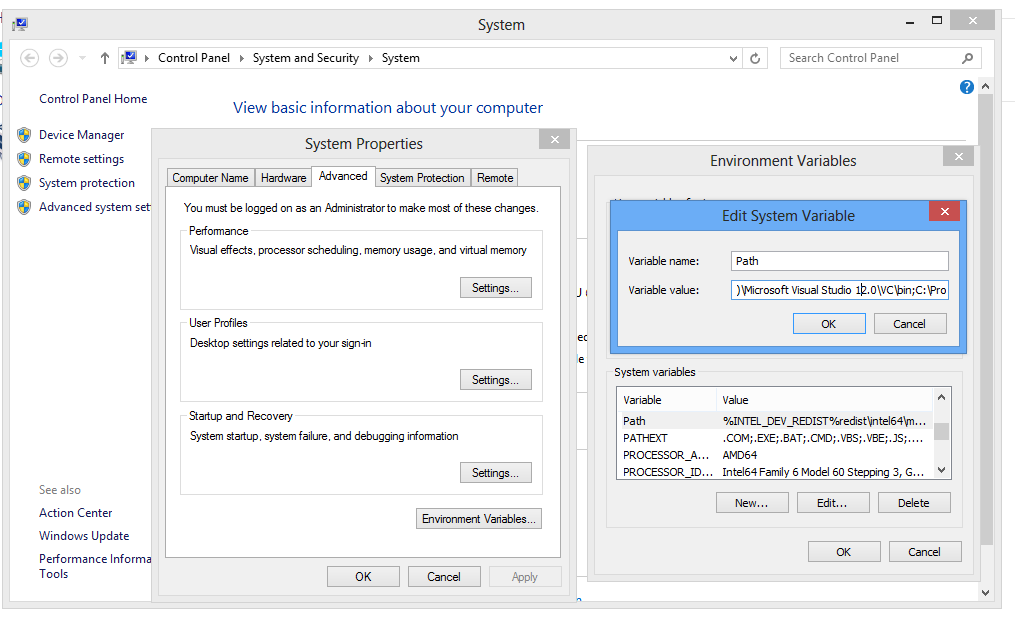
Only 64-bit versions of Windows 7 and Windows 8 are supported.

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:

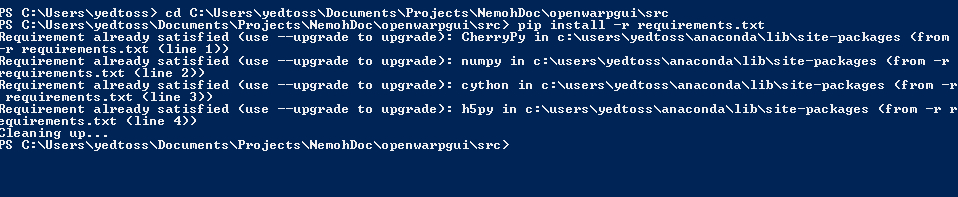


* Copy **$ROOT**\src\bundled\simulation\libs\libnemoh.dll, **$ROOT**\src\bundled\simulation\libs\libnemoh.dll.a, **$ROOT**\src\bundled\simulation\libs\libblas.dll, and **$ROOT**\src\bundled\simulation\libs\liblapack.dll to **$MINGW\_ROOT**\lib.
* Download and install Anaconda 2.10 with Python 2.7 for 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.

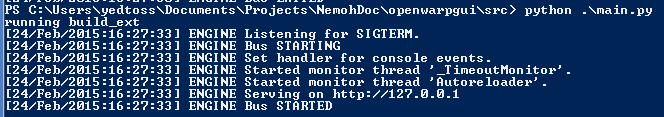
* Once the preceding steps are done, start PowerShell and install the CherryPy dependencies by executing this command in the **$ROOT**/src/ directory:

pip install -r requirements.txt



* Start the server by executing this in the src/ directory:

python main.py



* 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\_FORTRAN**" -G "MinGW Makefiles"

And finally run:

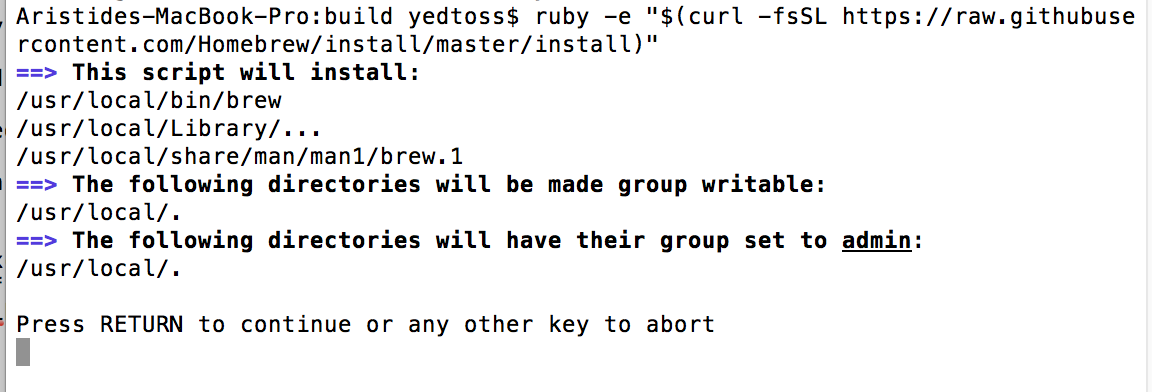
mingw32-make

# OS X DEPLOYMENT

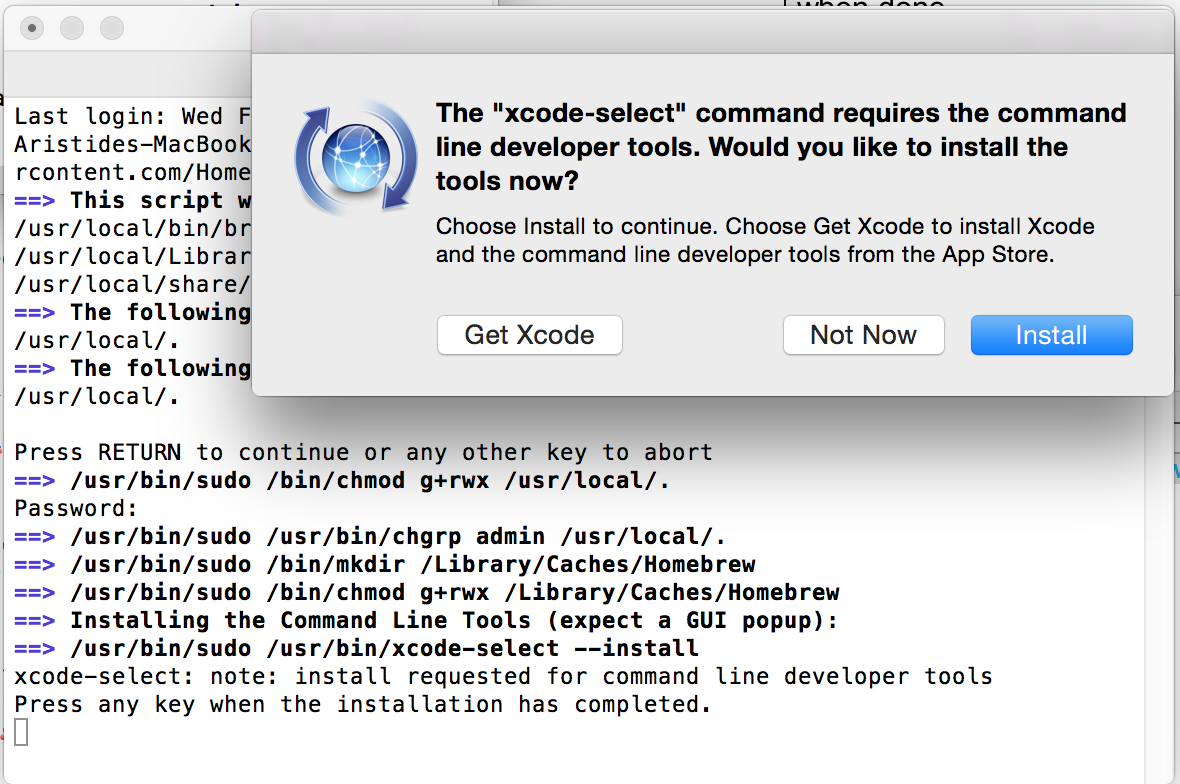
The following instructions are known to work on OS X versions 10.9 and 10.10.

To deploy the application in OS X, perform the following step:

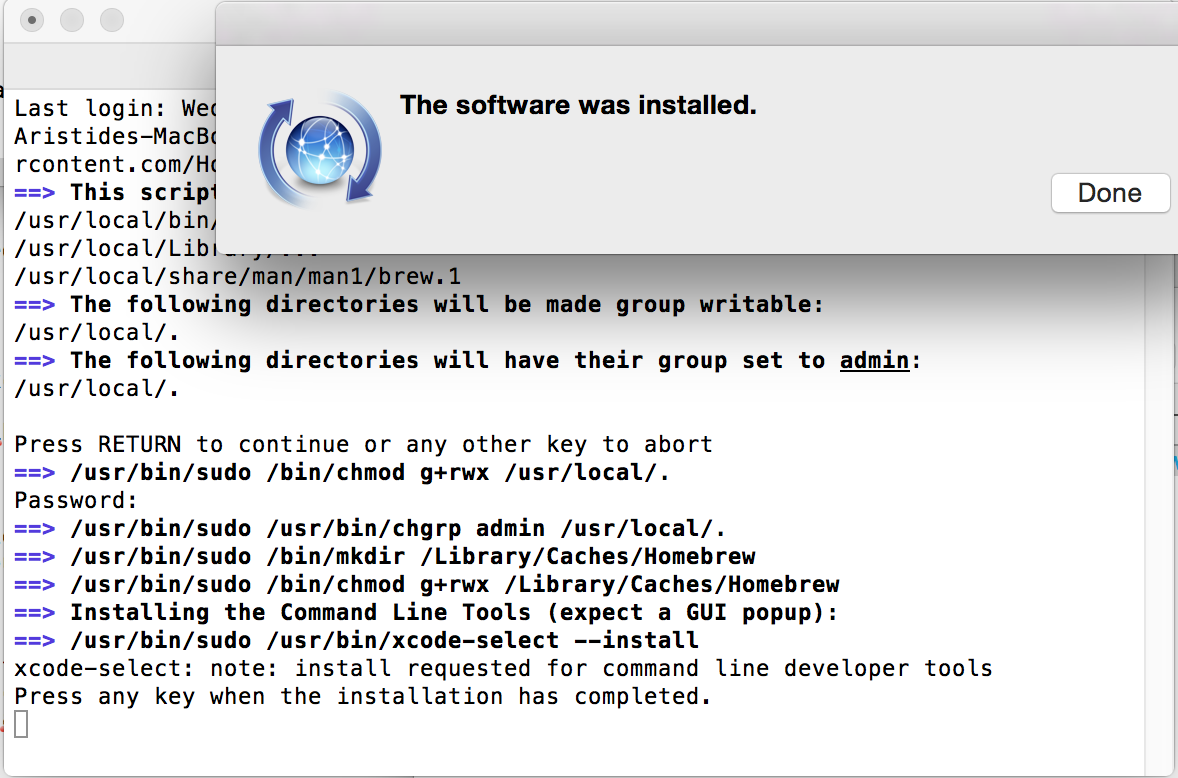
* Install brew: ruby -e "$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/master/install>)".



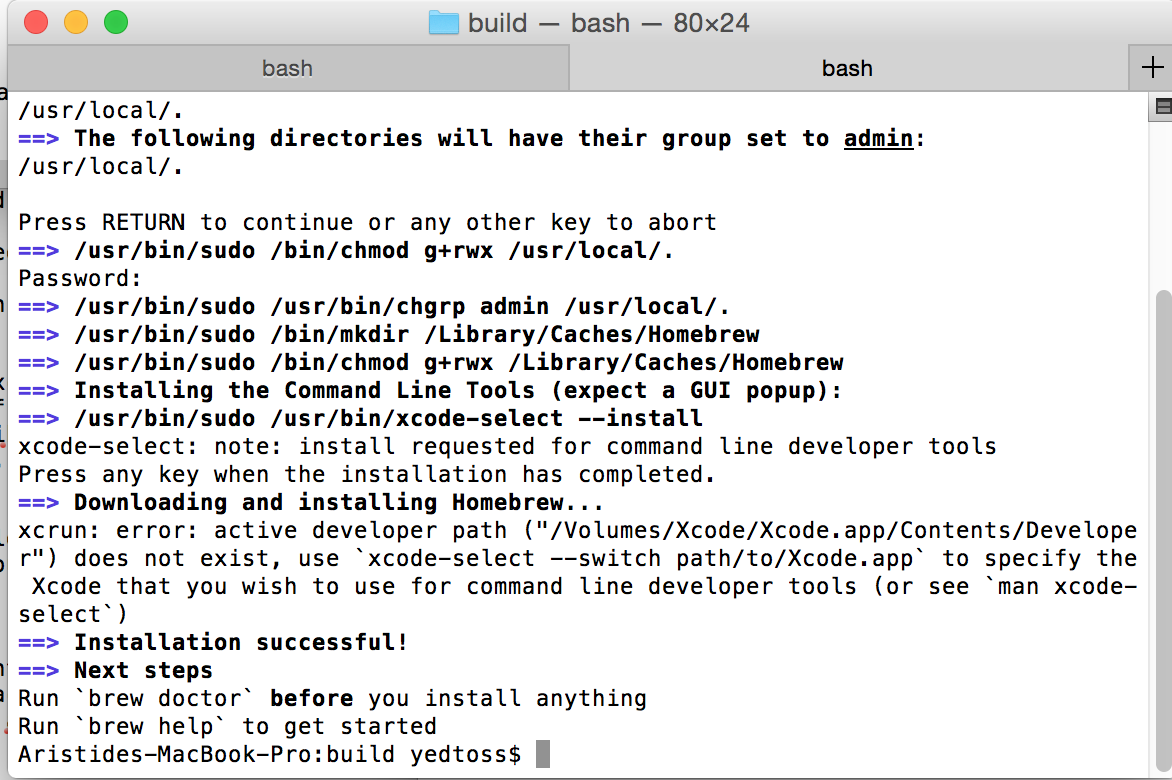
If you do not have the Xcode command-line tools, you will be prompted to install them as shown below.



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

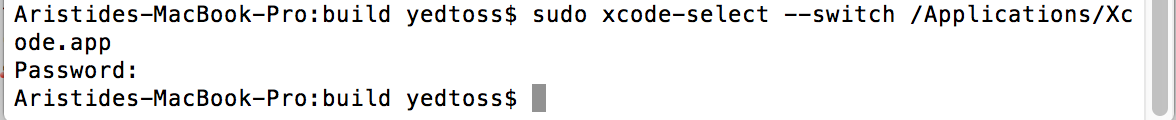


Press any key. It is possible that you will see an error about the active developer path:



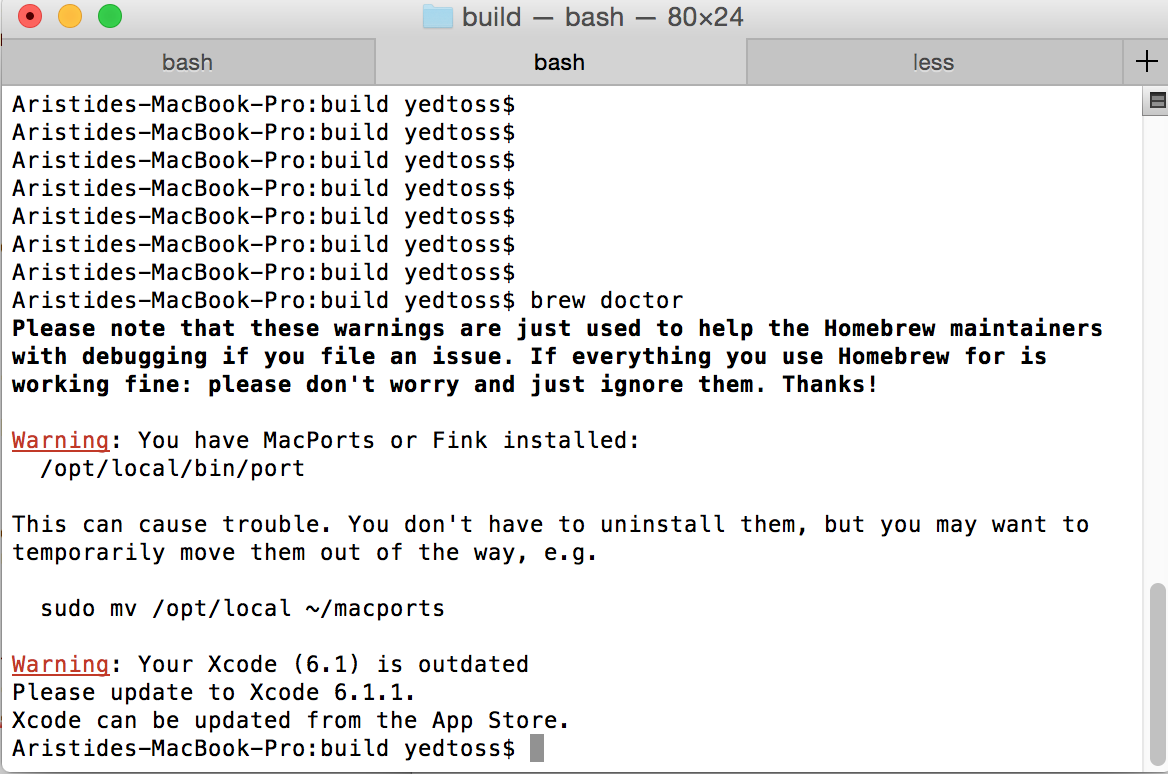
If you do get that error, you can correct it by specifying the version of Xcode to use:

sudo xcode-select --switch /Applications/Xcode.app



Now run this command:

brew doctor



Ignore any warnings that may appear.

Install GCC and GFortran:

brew install gcc

You can ignore warnings about multilib.

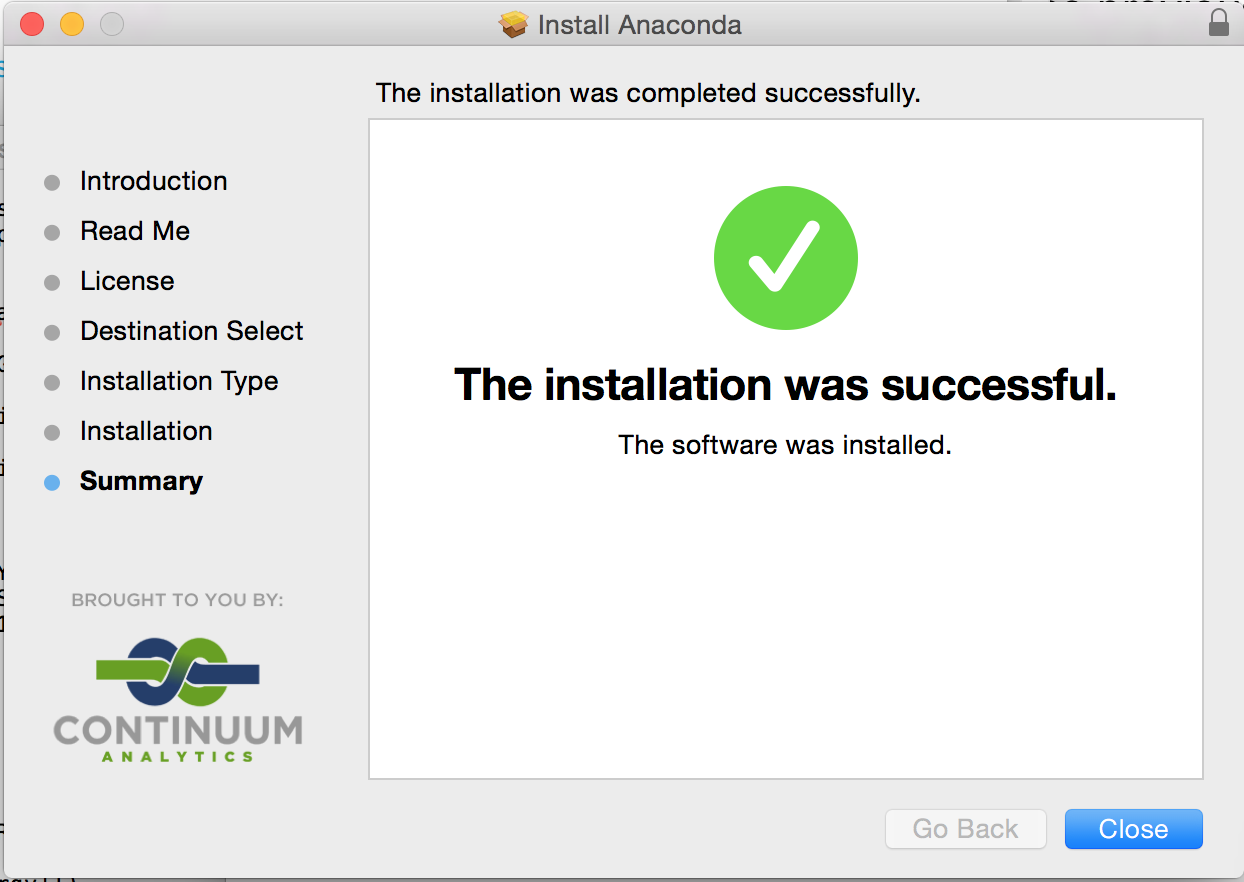


Install CMake:

brew install cmake



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:

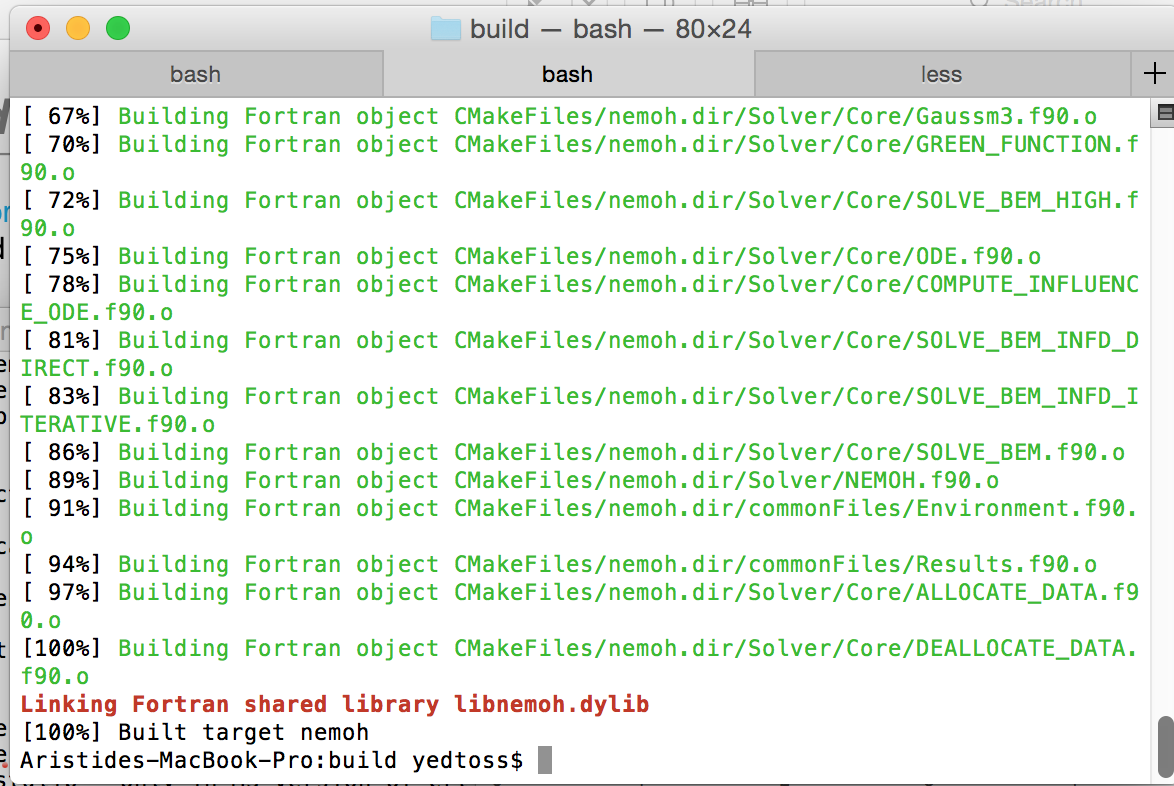
cmake -DCMAKE\_Fortran\_COMPILER="gfortran" **$NEMOH\_FORTRAN**

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.



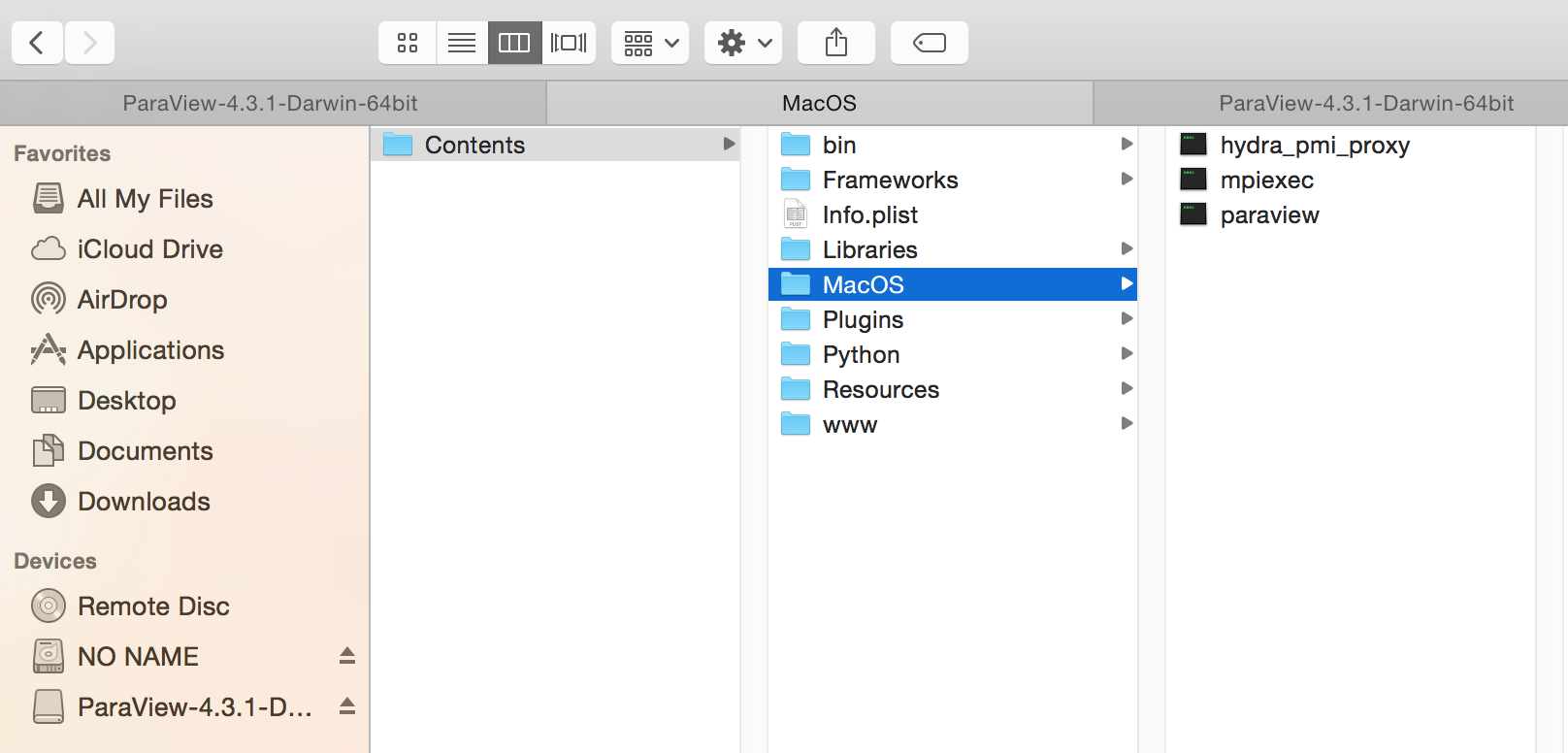
* Copy libnemoh.dylib from **$FORTRAN\_BUILD** to the lib/ directory inside the Anaconda installation root:

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:

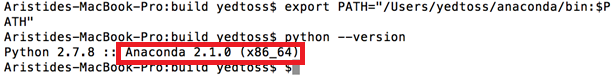
export PATH=/Users/bob/anaconda/bin:$PATH

(Replace "bob" with your own user name.)

Now run:

python --version

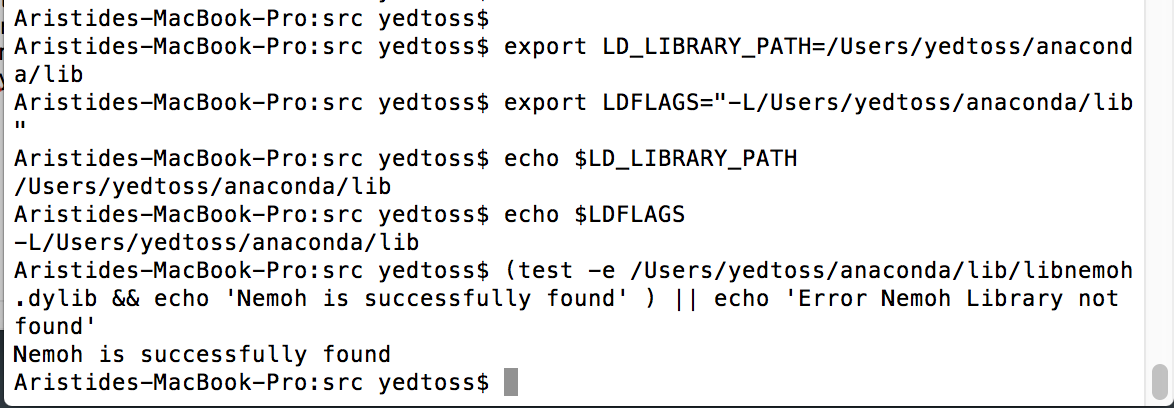
You should see Anaconda in the output as shown here:



* Run the following commands 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.)



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

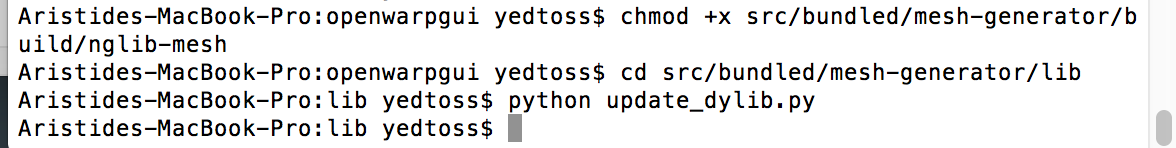
* Make sure the nglib-mesh binary is runnable:

chmod +x **$ROOT**/src/bundled/mesh-generator/build/nglib-mesh

(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:

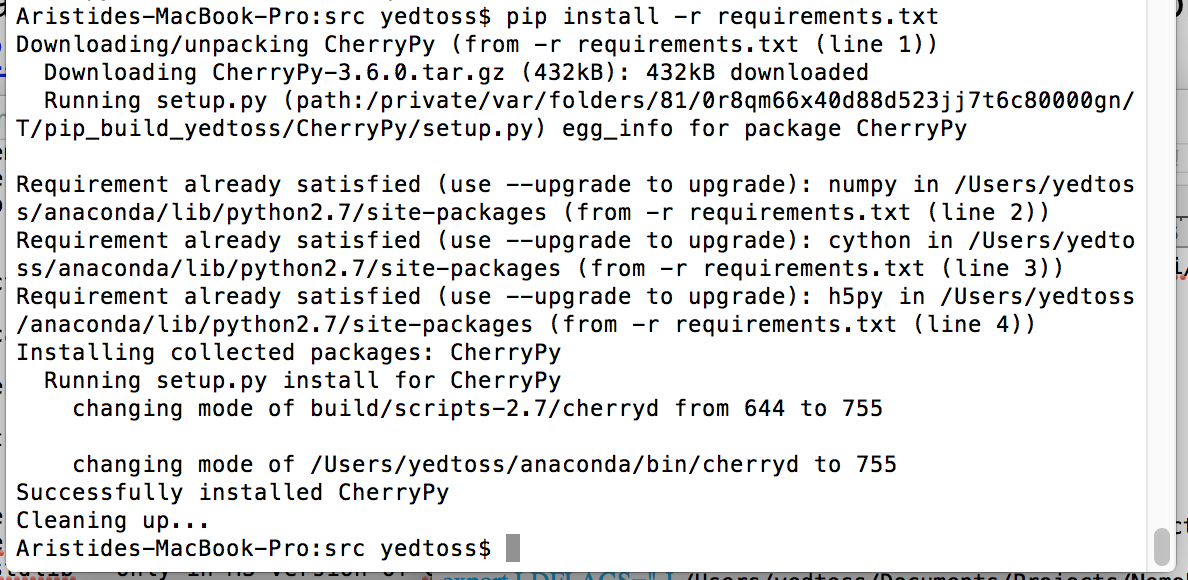
python update\_dylib.py



* Install the remaining Python dependencies by going to **$ROOT**/src/ and running:

pip install -r requirements.txt

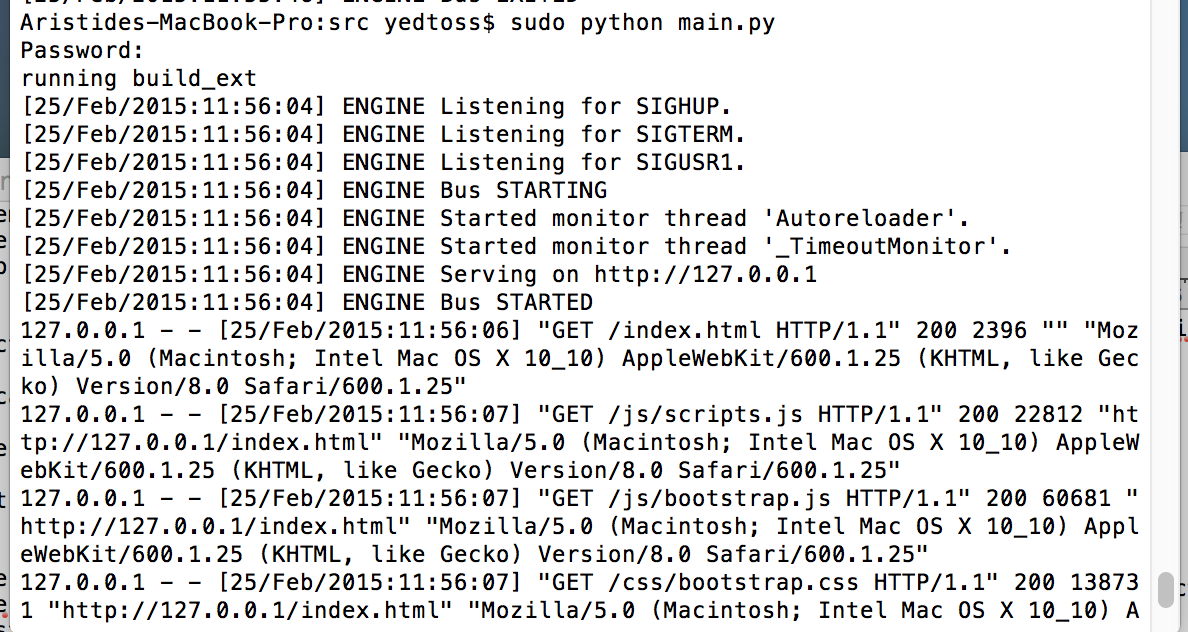
This step is required before running the OpenWarp server for the first time. You can skip it when running the server on subsequent occasions.



* 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.



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.

Note that in the above process we did not explicitly install the LAPACK or BLAS libraries. This is because they are installed by the brew command together with the Xcode Command Line Tools. If for some reason you receive an error when linking against LAPACK or BLAS, you can install a custom version by running:

brew install https://raw.githubusercontent.com/Homebrew/homebrew-dupes/master/lapack.rb

# 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.

## 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.

## 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.

## 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.

# USAGE

## Start the application

Make sure the server has been started:

python main.py

This command must be executed from the **$ROOT**/src/ directory, where **$ROOT** is the top-level directory of the OpenWarp package.

Open a web browser and go to:

<http://127.0.0.1/index.html>

You will see the following.



You can quit the application by clicking the orange Quit button.

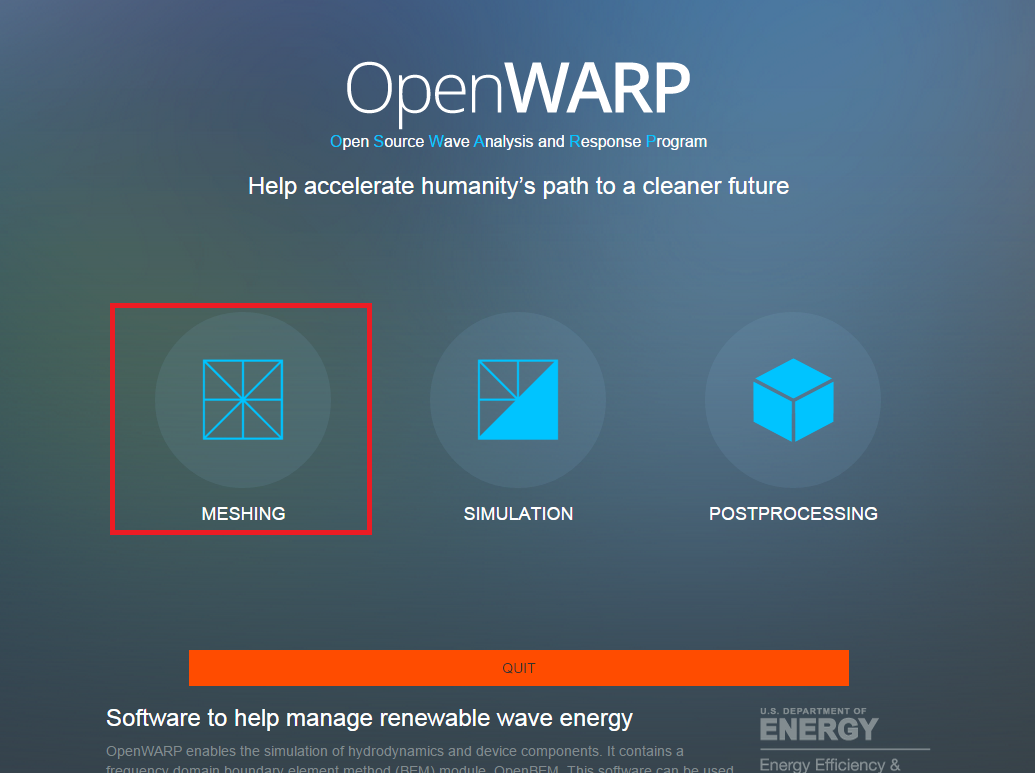
Quitting results in:



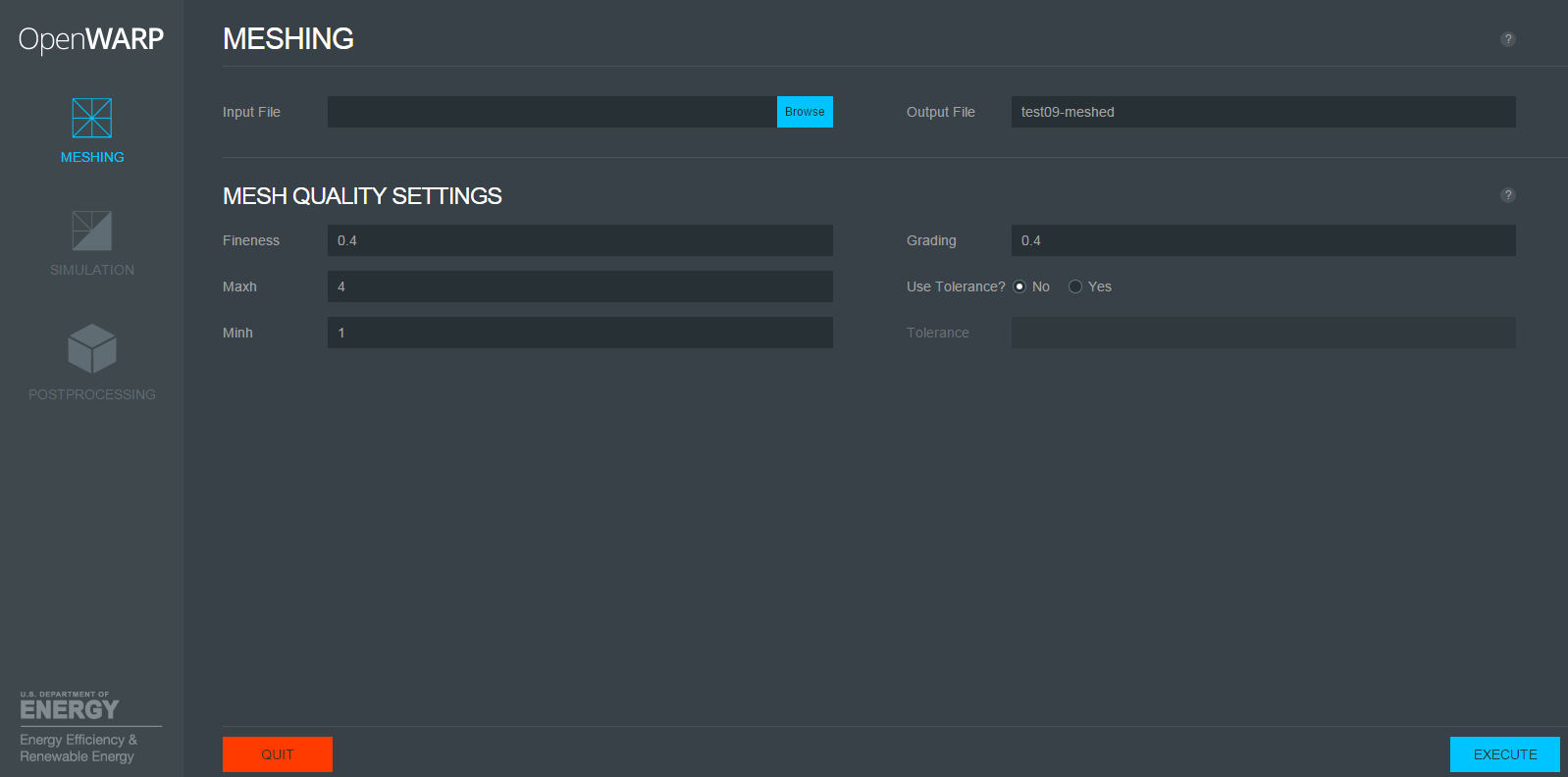
You must restart the Python server before using the application again.

## Start meshing

Click the Meshing button:



This is the meshing page:



Select the input file by clicking Browse and selecting a file.

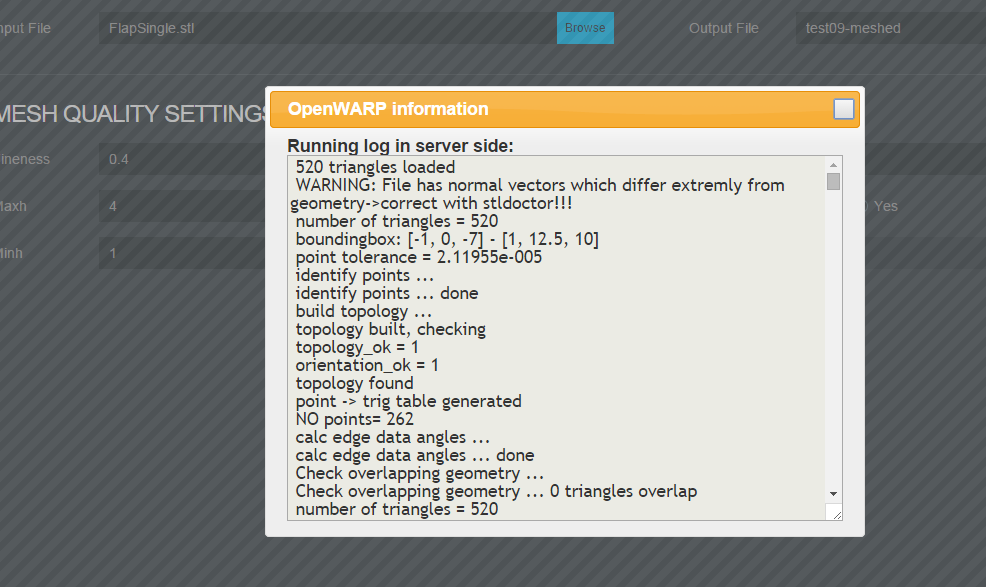
Set the other parameters according to your needs.

## Execute meshing

Click the blue Execute button.

## 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 salso convert 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:

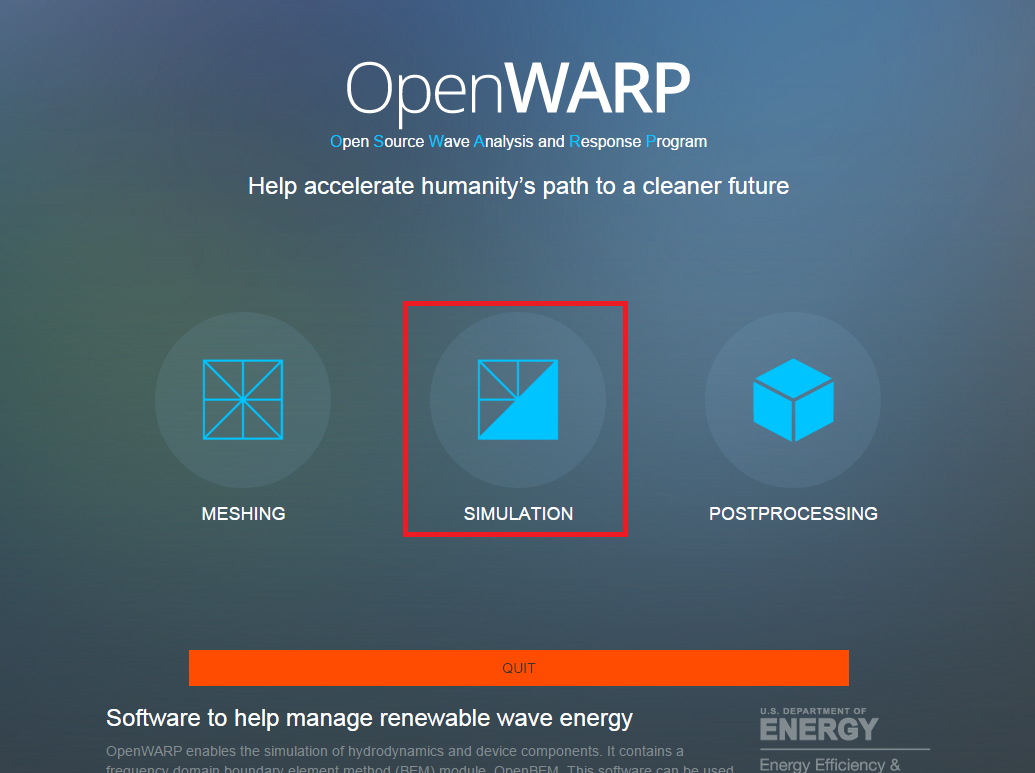
* Two files ending in .vtp, containing the mesh in VTP format. The "quad" version contains only quadrilateral elements.
* Two files ending in .vtk, containing the mesh in VTK format.
* Two files ending in .dat, the format recognized by Nemoh.
* Two files ending in .gdf, containing the mesh in GDF format.
* 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.

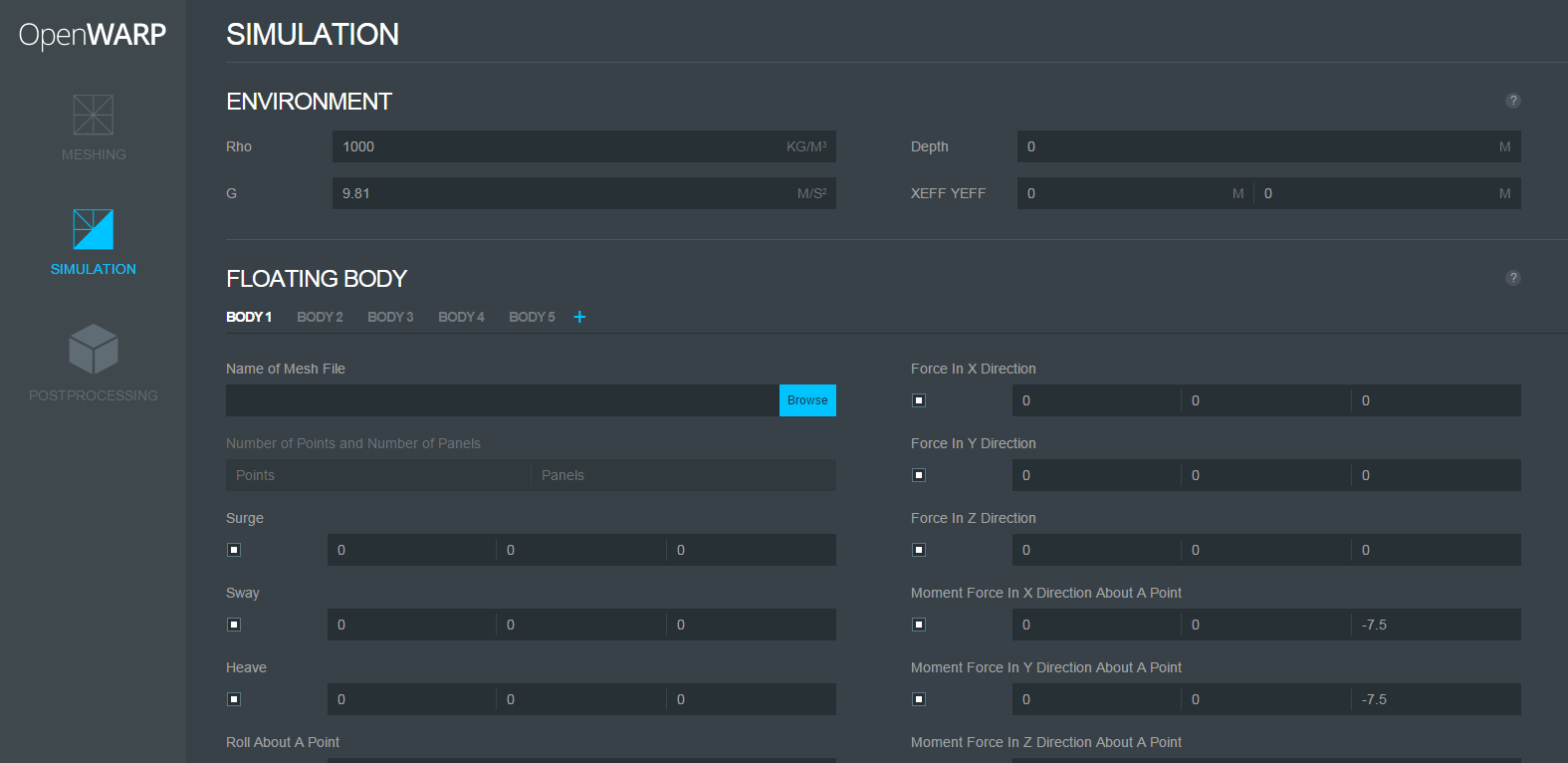
For each pair of files, the "quad" version contains only quadrilateral elements.

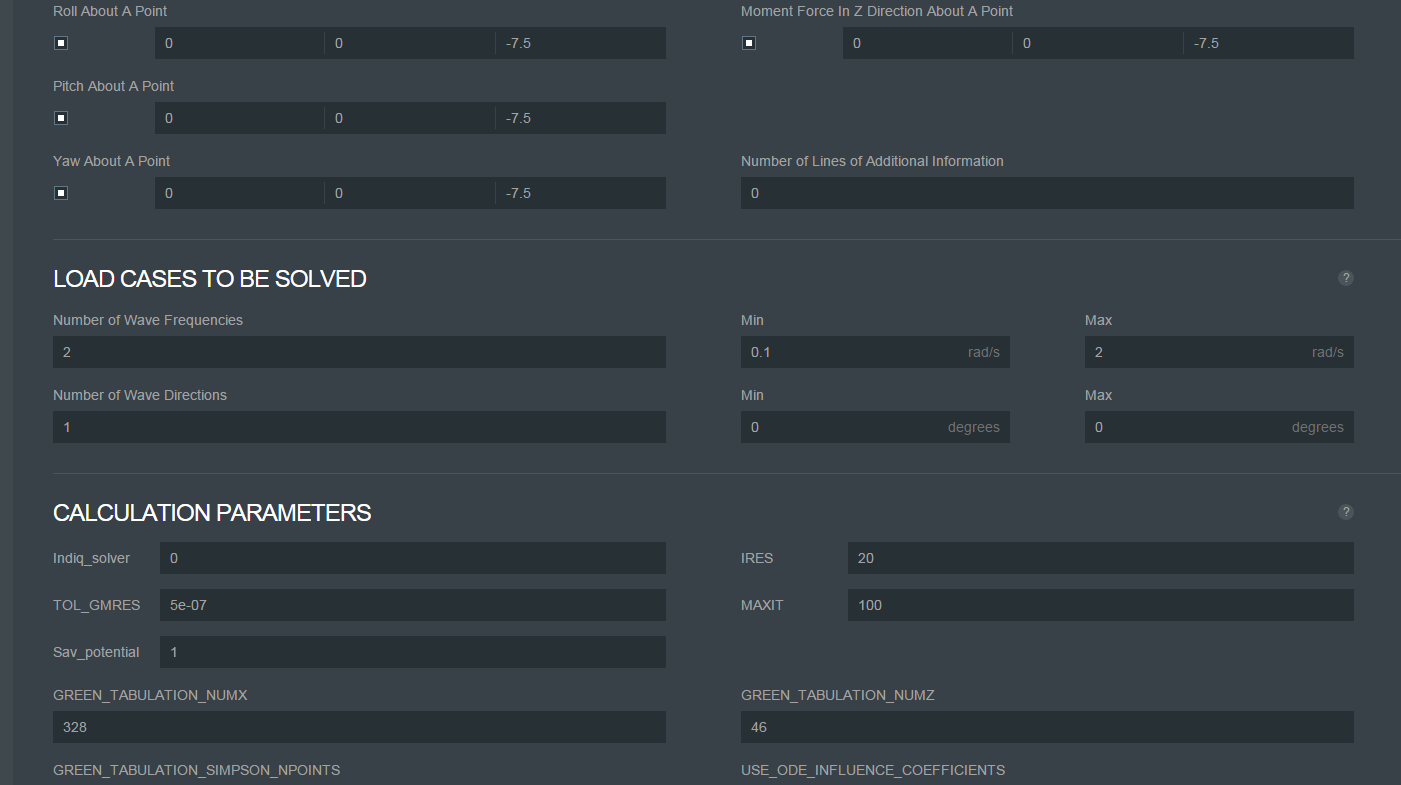
## Start the simulation

Click the Simulation button:



This is the resulting page:

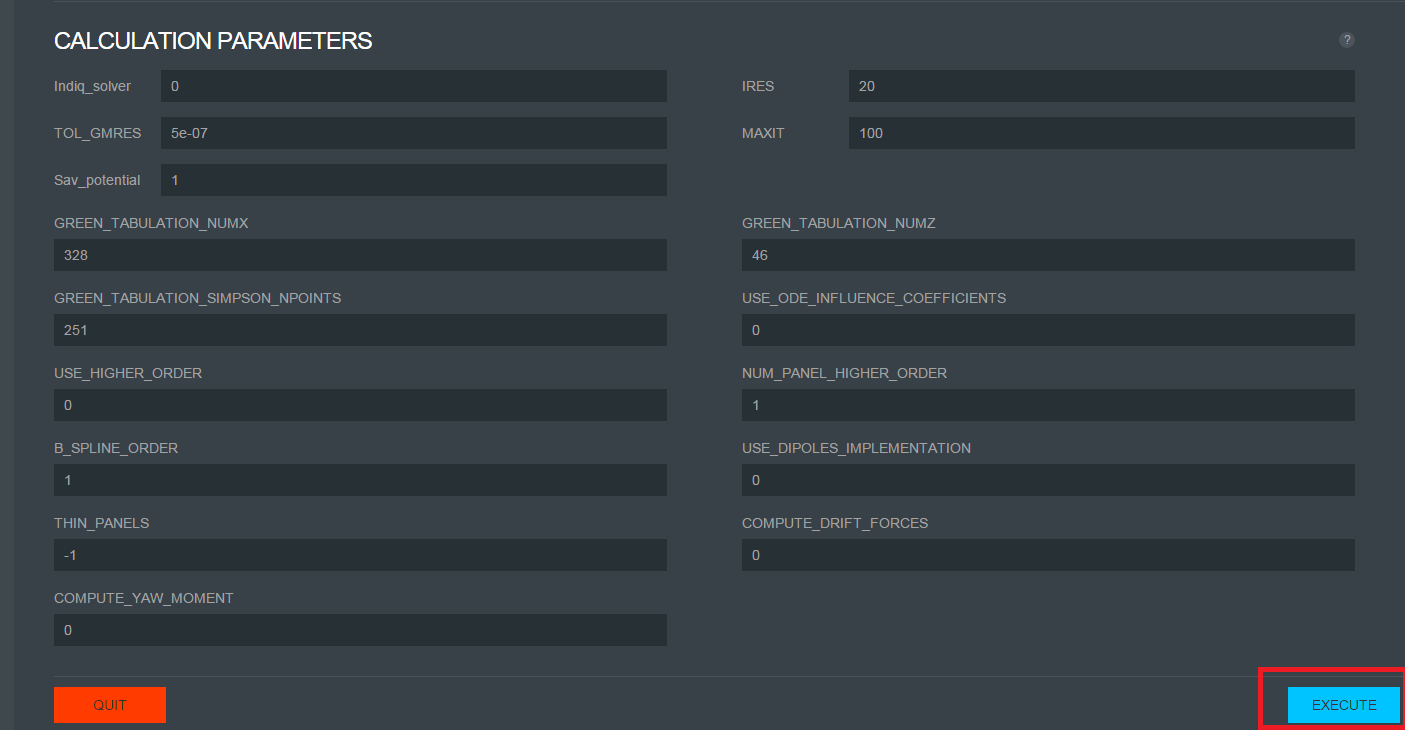






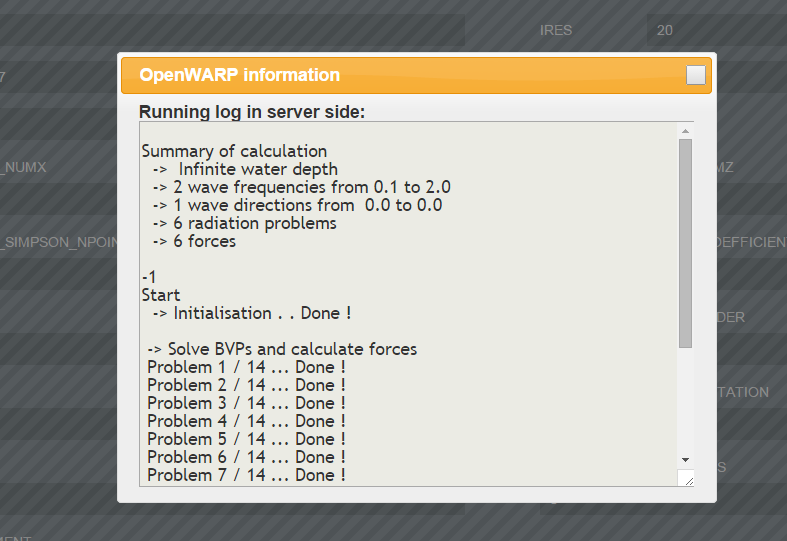
## Execute the simulation

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



## 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.

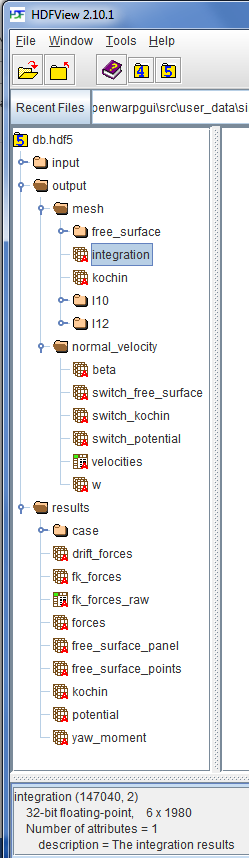
The timestamp is in the format YYYYMMDDHHMM.

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:

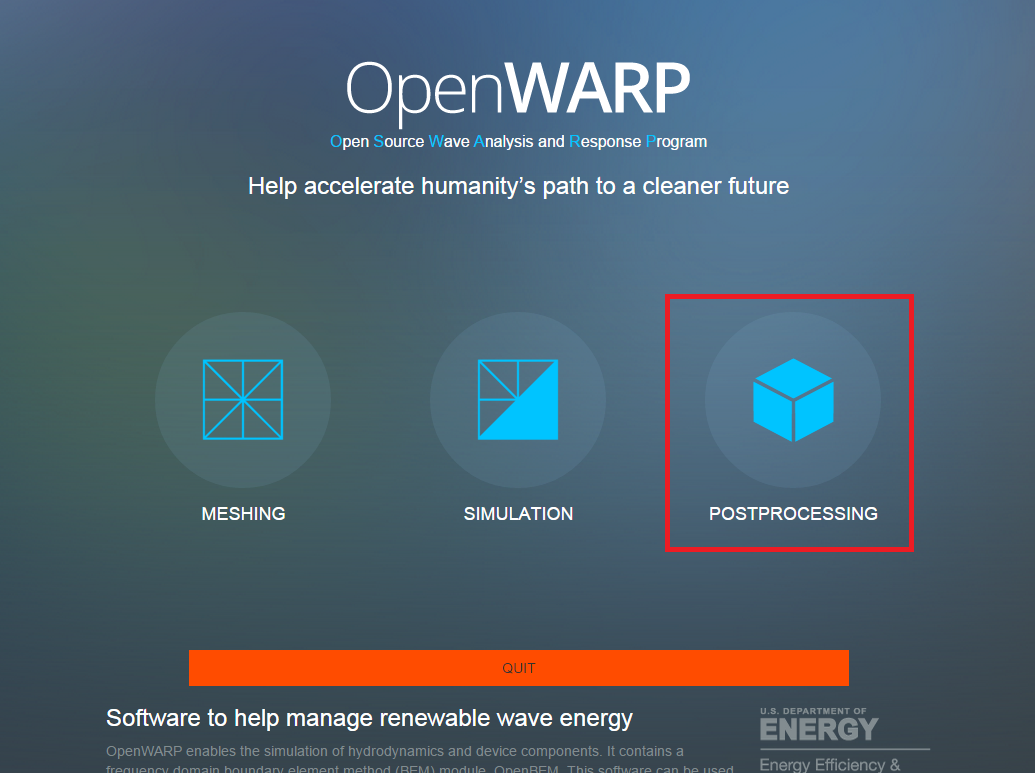


You will see that it contains three main groups: the input group, the output group containing intermediate results generated by the application, and the results group containing final results. In each group, you can see datasets with their description as shown in the figure below.

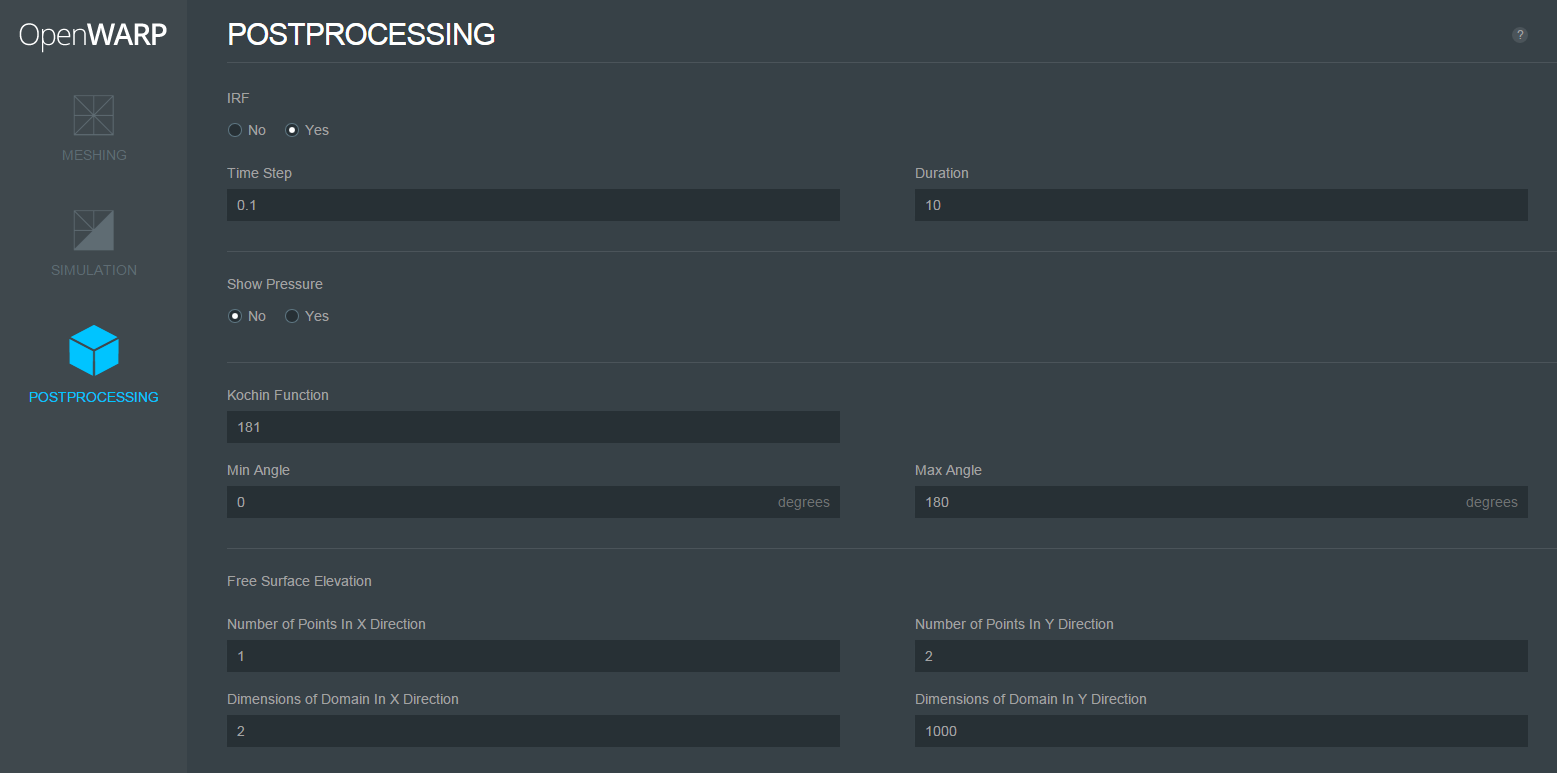


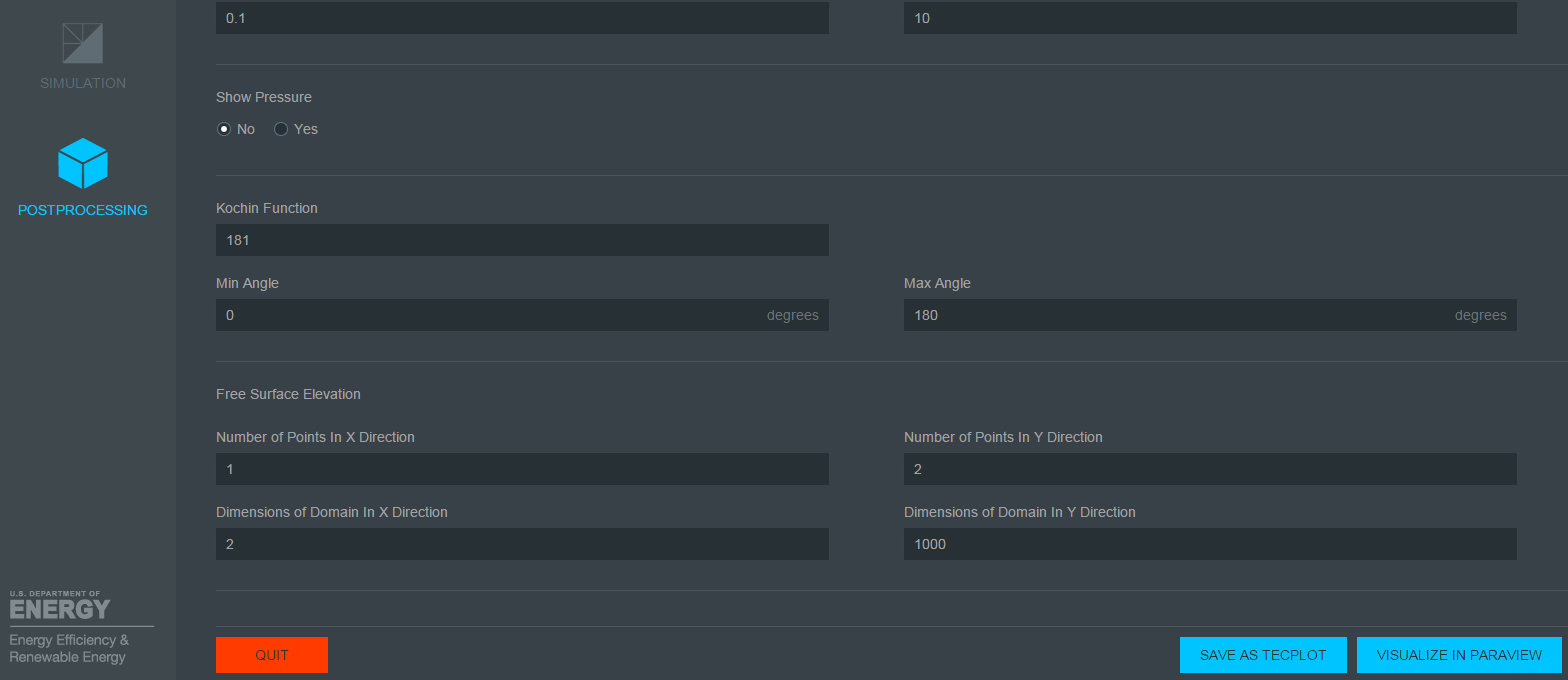
## Postprocessing

Click the Postprocessing button:



You should see the following:





## 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.



## 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 YYYYMMDDHHMM.

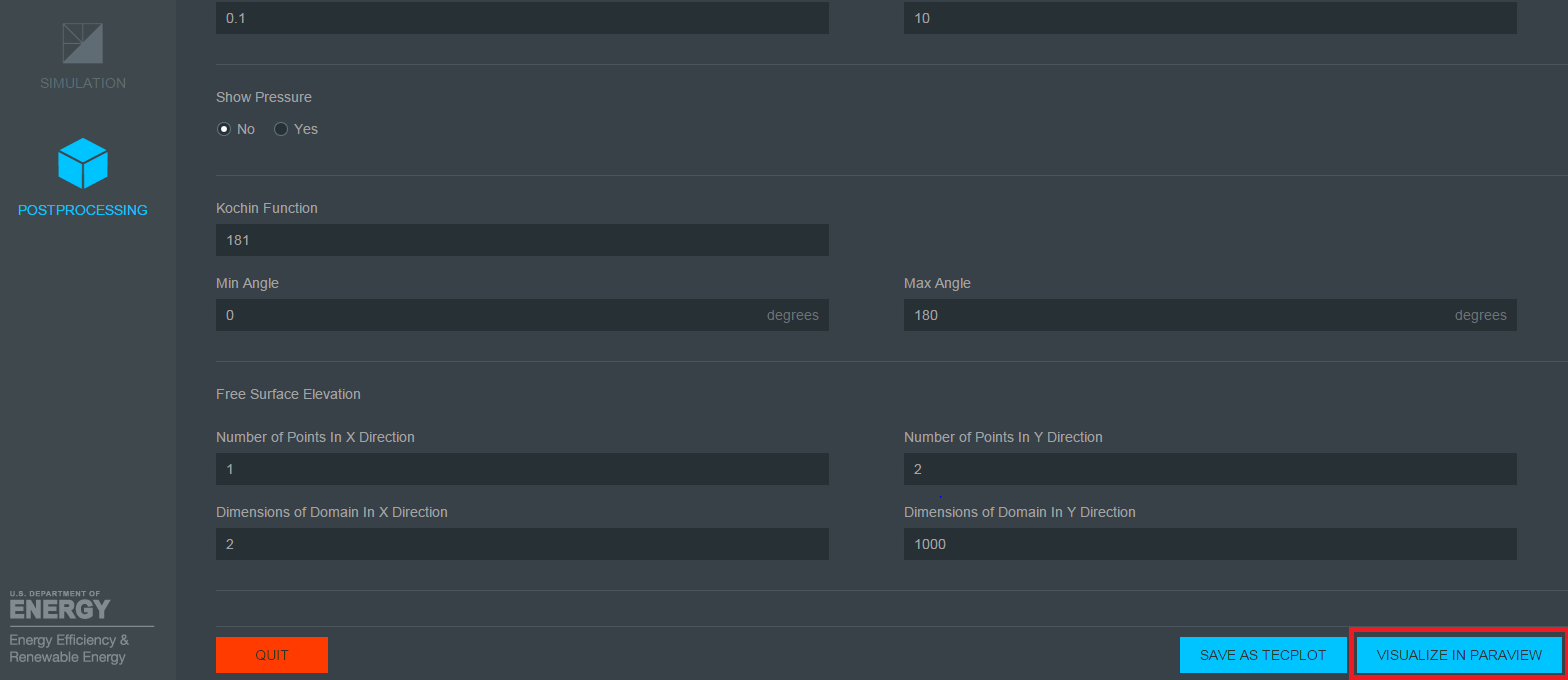
Navigate to the directory with the latest timestamp.

You will find the following output files:

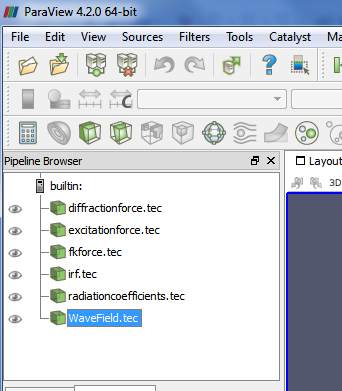
* results/irf.tec: This file contains the infinite frequency added mass and the impulse response function for the radiation force.
* results/WaveField.tec: The wave field in TECPLOT format.
* results/diffractionforce.tec: The diffraction force for the diffraction problems.
* results/excitationforce.tec: 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.

## 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":

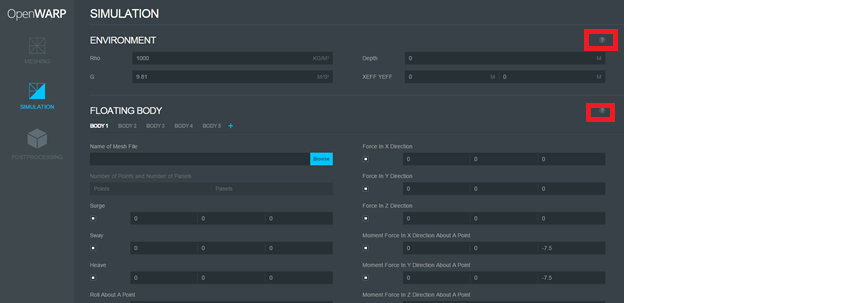


ParaView will start and you will find the resulting forces on the left side:



## Getting help

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



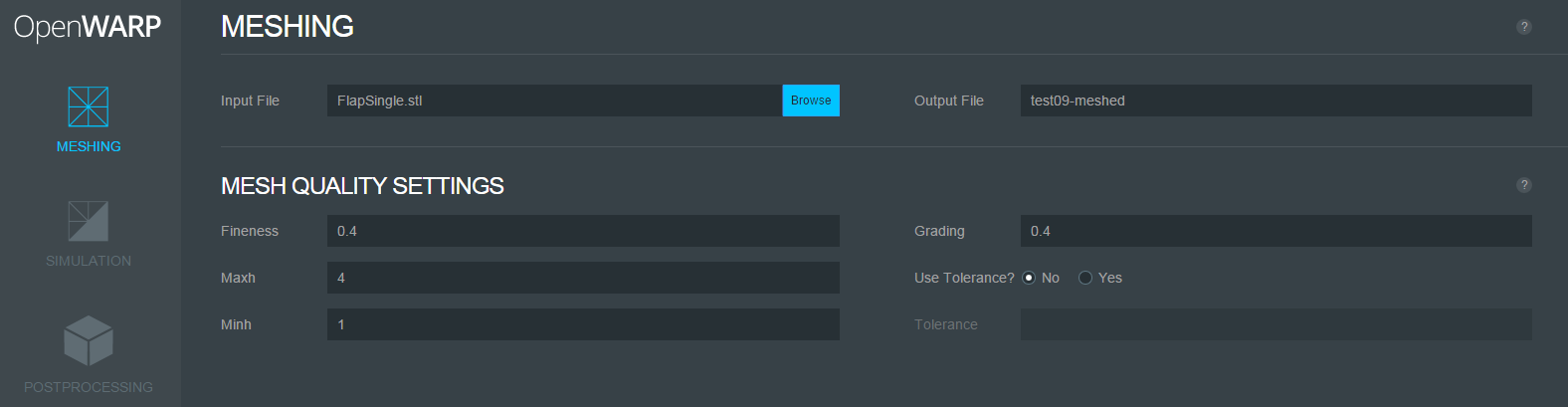
# EXAMPLE

We are going to use the FlapSingle example available in **$ROOT**/test\_files where **$ROOT** is the top-level directory of the OpenWarp package.

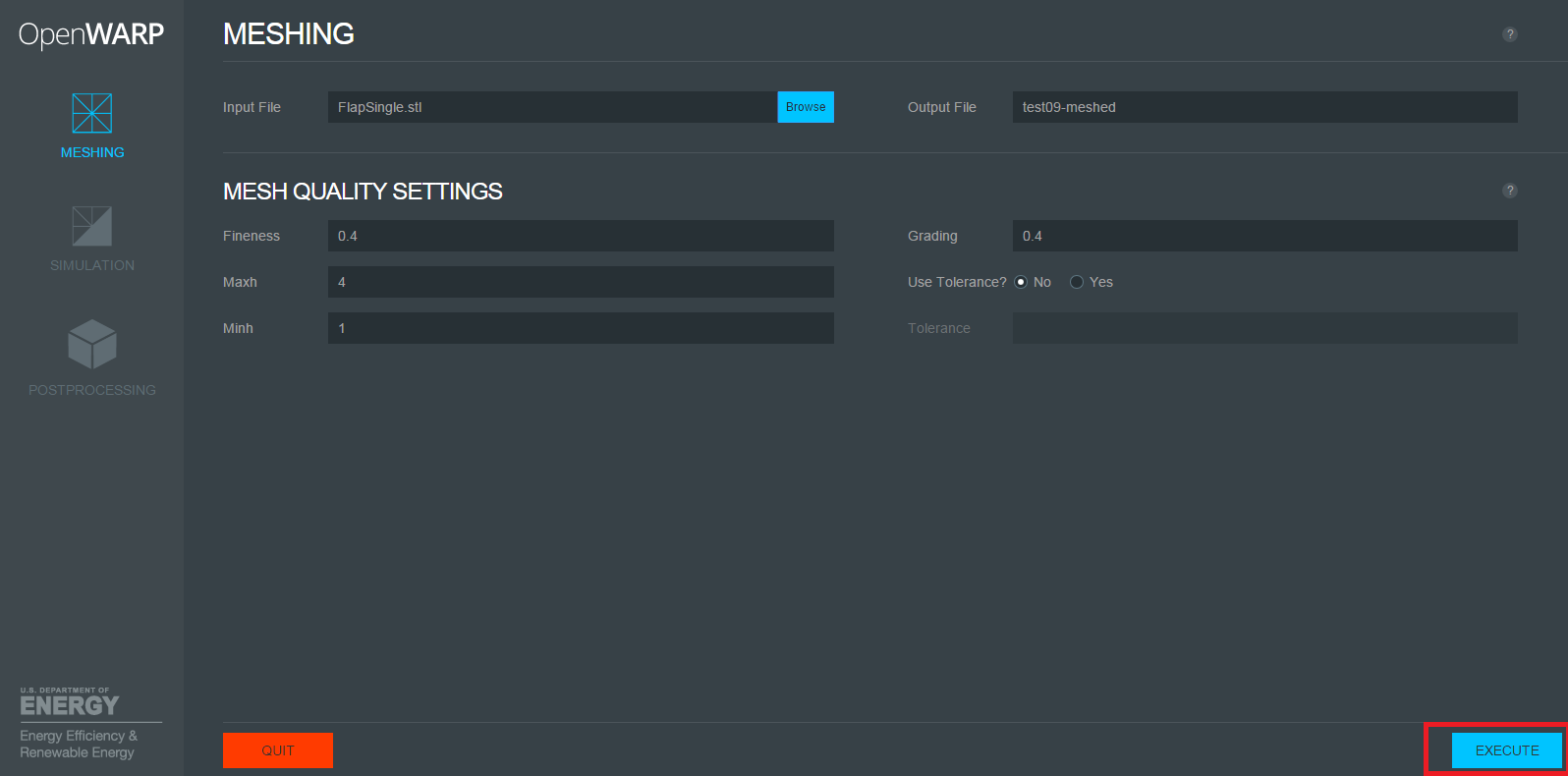
First, we will generate the mesh of the body. Next, we will run the simulation. Finally, we will generate the TEC files and visualize them.

## Mesh Generation

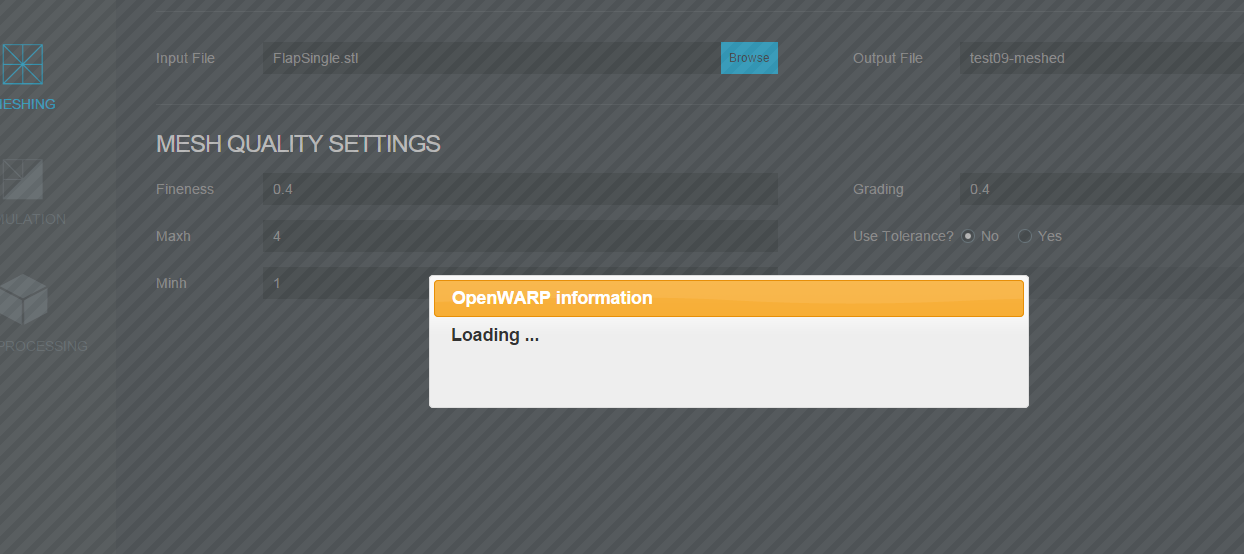
Go to the Meshing page, click on Browse, and choose **$ROOT**/test\_files/mesh/FlapSingle.stl where **$ROOT** is the top-level directory of the OpenWarp package.



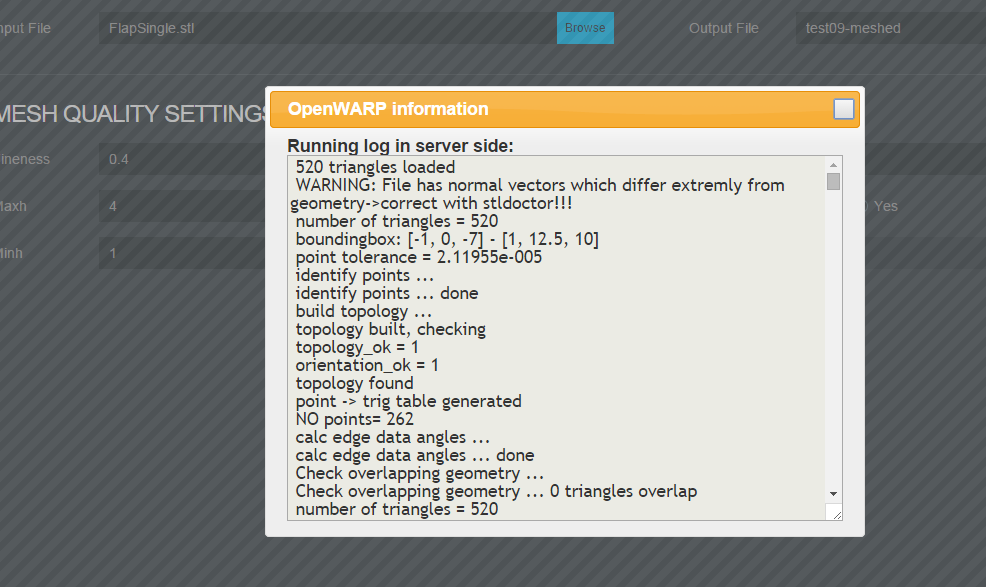
Execute the meshing:



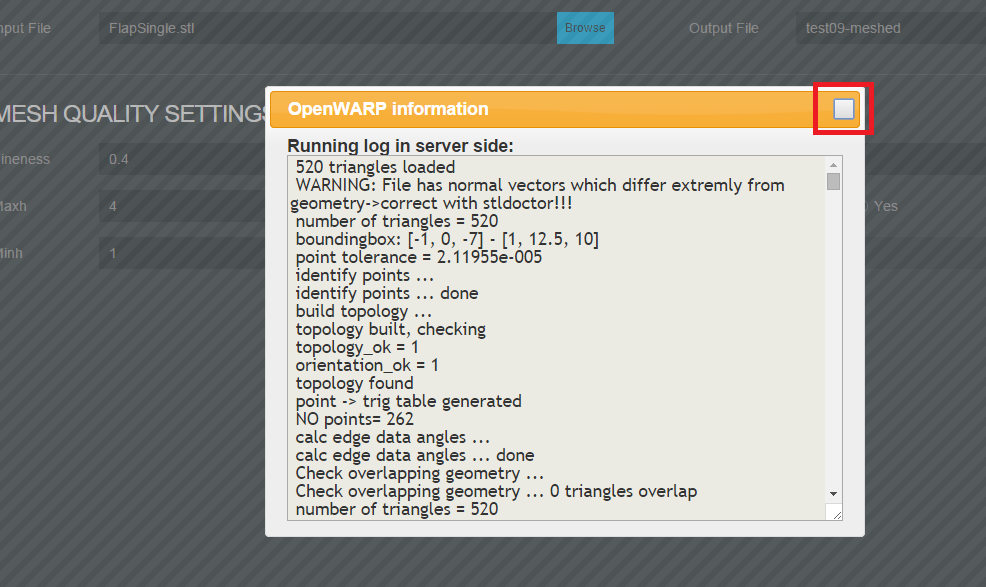
You will see:



The results are shown like this:



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

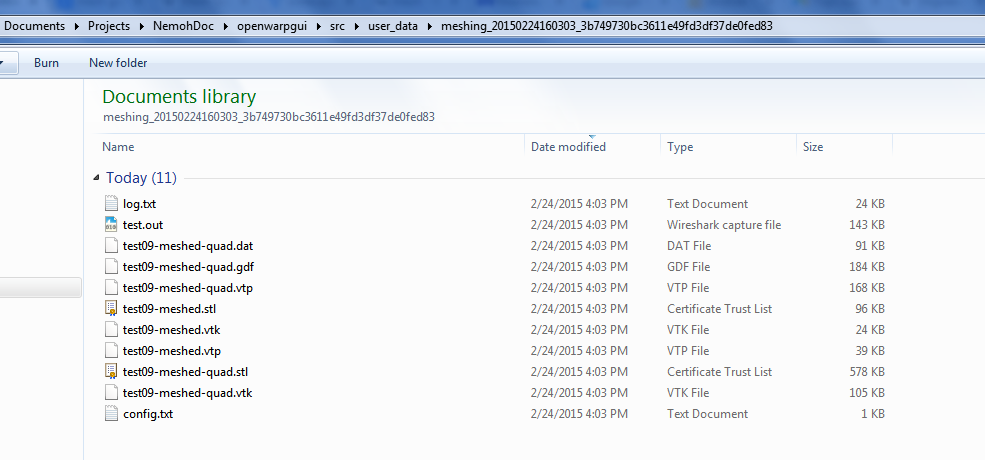


## Run the simulation

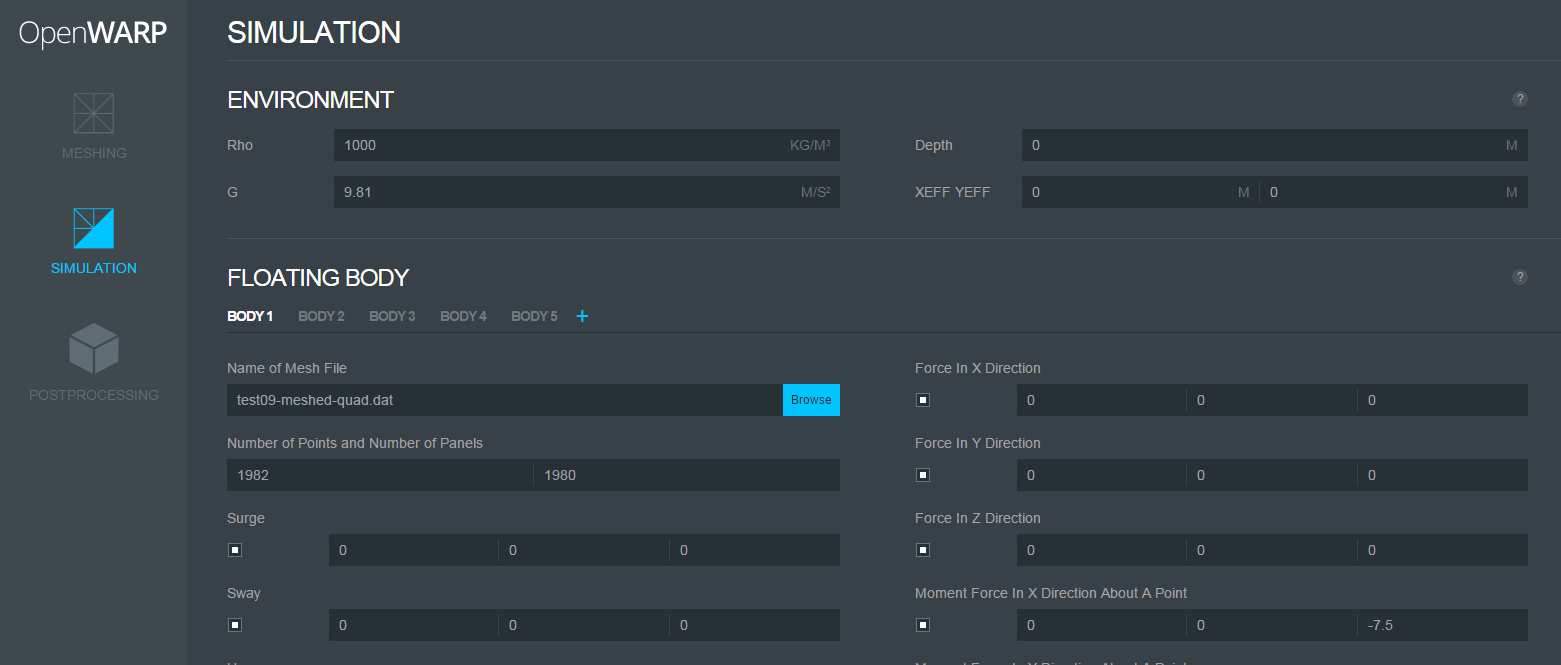
Go to the simulation page and click on Browse to load the generated mesh file.

The generated mesh file is inside the **$ROOT**/src/user\_data/meshing\_TIMESTAMP\_HASH where **$ROOT** is the top-level directory of the OpenWarp package.

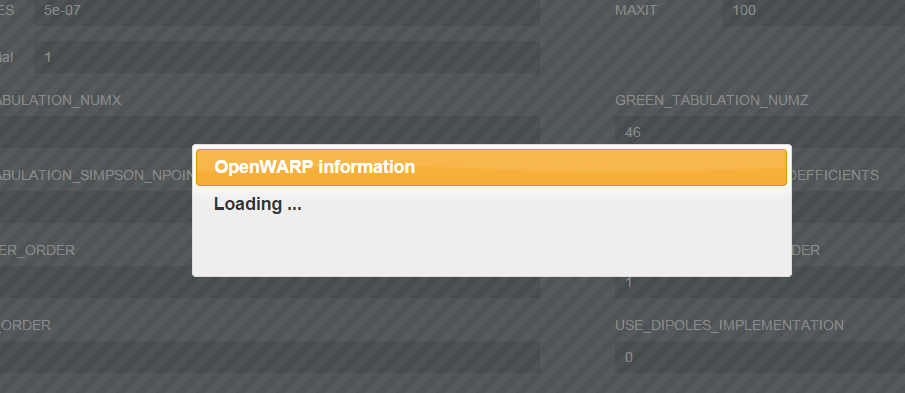
Go to the directory with the latest timestamp.



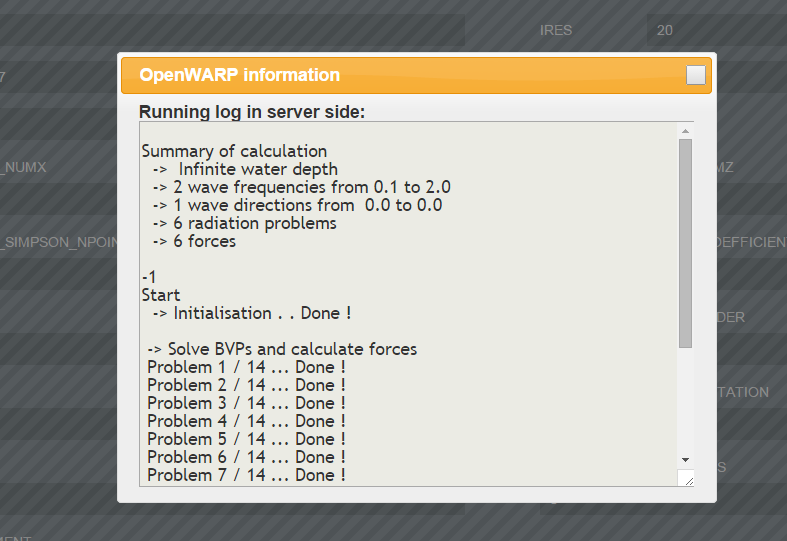
Look for a .dat file. It is named test09-meshed-quad.dat in our case.



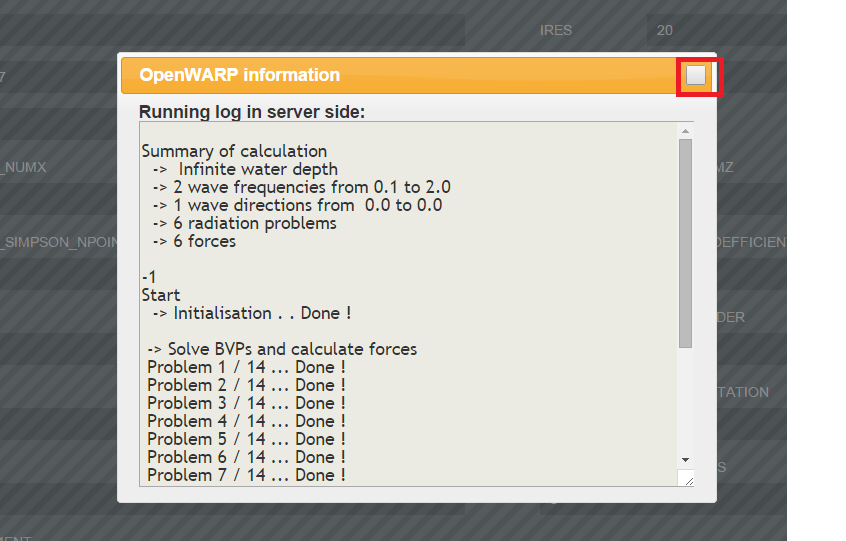
Run the simulation. You should see this:



When the simulation is done, the results are summarized in a popup:

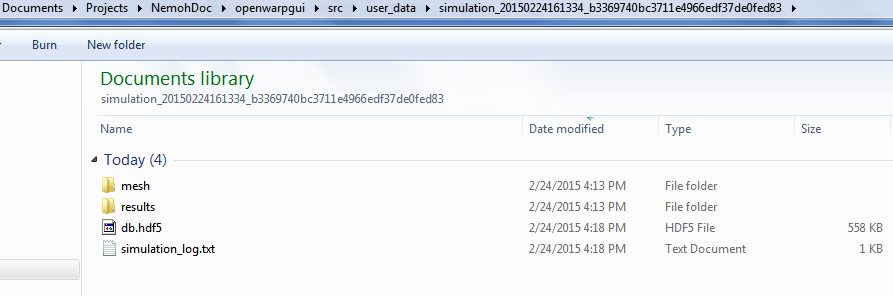


Close the popup:



The generated output directory is **$ROOT**/src/user\_data/simulation\_TIMESTAMP\_HASH where **$ROOT** is the top-level directory of the OpenWarp package.

Find db.hdf5 inside the directory with the latest timestamp.

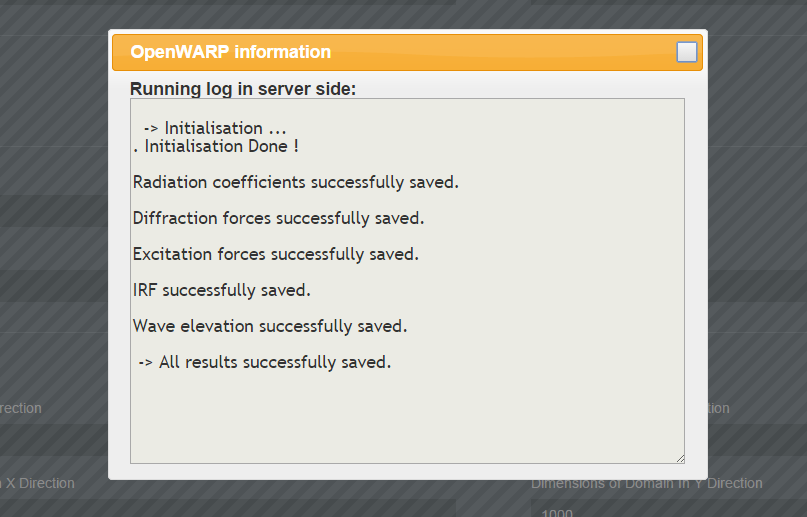


## Generate TEC files and run postprocessing

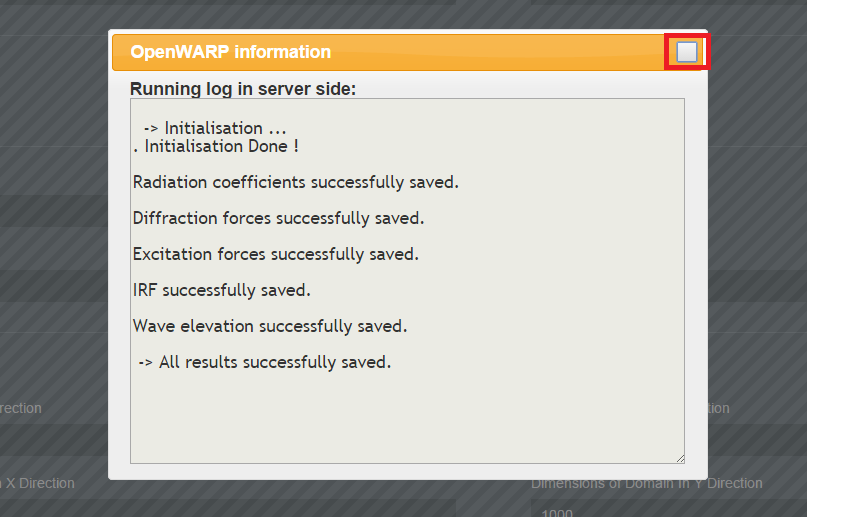
Open the postprocessing page.

Leave the default values. Generate the TEC files by clicking "Save as TECPLOT".

When postprocessing is done, you get a popup:



Close it.



## 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:

