1- How the bugs were solved

a) Why the code was failing

We first try to make the bugs checkable consistently. Due to the fact that it was inconsistent and there were no logs about what went wrong, we assumed it was due to two bugs:

Bug #1 There is a segmentation fault in the fortran code or in the python code

Bug #2 The standard output and error of the functions run in a child process were not showing consistently in the main process

b) Making the code fail consistently

As a segmentation fault is usually caused by uninitialized array, variables or access to array indices outside its size, we modified to debugs flags of the fortran code in OpenWARP/source/NemohImproved/Nemoh/CMakeLists.txt to make it looks like:

```
set(CMAKE_Fortran_FLAGS_DEBUG "-Wall  -std=f2008 -fopenmp   -O0  -g  -Wextra -fimplicit-none -fbounds-check -fbacktrace “)
```

The -fbounds-check means that an error will be reported and the program stop if we are accessing an array out of bounds. It also means any access to a non initialized variable will make the program failed and report an appropriate error message

The -fbacktrace is to print the full trace in case of any error

We also removed the -W and -fmax-errors=0 to show all compilation warnings.

We finally recompiled the fortran module in Debug mode using:

```
cmake -DCMAKE_Fortran_COMPILER="gfortran" $NEMOH_FORTRAN -DCMAKE_BUILD_TYPE=Debug
```

After doing this the code was consistently failing no matter which value we were choosing. But still we got no logs about the failure

c) Solving Bug #2

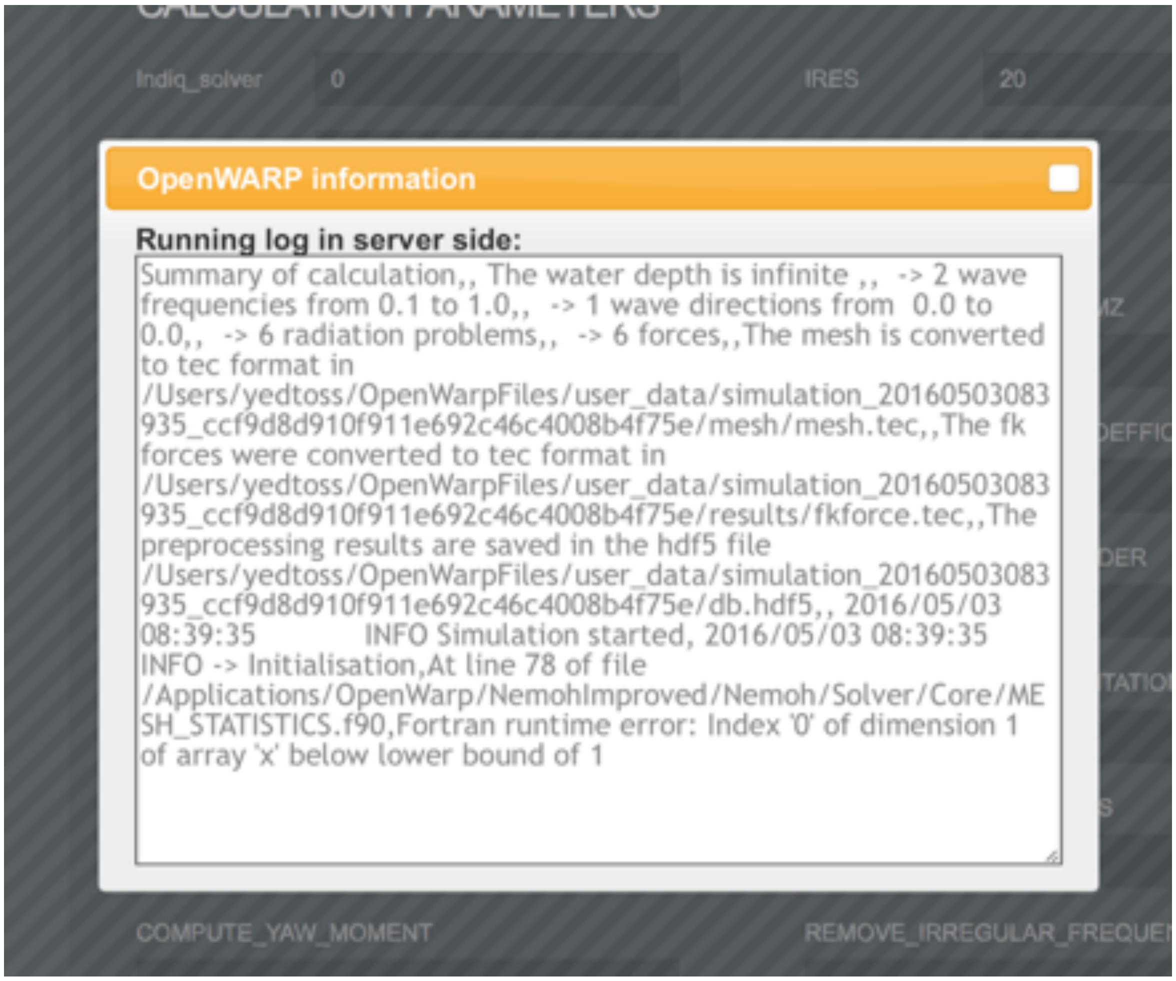
To be able to do anything we need to see the error message. The preprocessor and solver need to be run in a subprocess to avoid the server failing in case of error. Then, we need to get the standard output and error from this separate process. The previous code was using a technique that is not consistent and would fail in case of error from the subprocess.

We switched to using the python module capturer. In the solver, we released the GIL before calling the fortran code to avoid polluting it in case the fortran code fails. Finally, we make sure there were no race conditions in the logging by making sure the subprocess does not log to the same file as the main process (something not supported by python).

After that we could see the detailed error message.

d) Solving Bug #1

The first error we saw is: “At line 78 of file /Applications/OpenWarp/NemohImproved/Nemoh/Solver/Core/MESH_STATISTICS.f90
Fortran runtime error: Index '0' of dimension 1 of array 'x' below lower bound of 1 (Coin(i, 1,j)=1.0*X(p(j, i)))” as shown below



Running log in server side:

```
Summary of calculation,, The water depth is infinite ,,  -> 2 wave
frequencies from 0.1 to 1.0,,  -> 1 wave directions from  0.0 to
0.0,,  -> 6 radiation problems,,  -> 6 forces,,The mesh is converted
to tec format in
/Users/yedtoss/OpenWarpFiles/user_data/simulation_20160503083
935_ccf9d8d910f911e692c46c4008b4f75e/mesh/mesh.tec,,The fk
forces were converted to tec format in
/Users/yedtoss/OpenWarpFiles/user_data/simulation_20160503083
935_ccf9d8d910f911e692c46c4008b4f75e/results/fkforce.tec,,The
preprocessing results are saved in the hdf5 file
/Users/yedtoss/OpenWarpFiles/user_data/simulation_20160503083
935_ccf9d8d910f911e692c46c4008b4f75e/db.hdf5,, 2016/05/03
08:39:35            INFO Simulation started, 2016/05/03 08:39:35
INFO -> Initialisation,At line 78 of file
/Applications/OpenWarp/NemohImproved/Nemoh/Solver/Core/ME
SH_STATISTICS.f90,Fortran runtime error: Index '0' of dimension 1
of array 'x' below lower bound of 1
```

The cause of this error is that the previous code was not correctly setting the 3rd and 4th dimension of the matrix p .
This was previously set in NEMOH.f90 using:

```
stat_p(stat_tmp1, 1) = stat_tmp2
stat_p(stat_tmp1, 2) = stat_tmp2 + 1
stat_p(stat_tmp1, 1) = stat_tmp2 + 2
stat_p(stat_tmp1, 2) = stat_tmp2 + 3
```

We thus changes it to

```
stat_p(stat_tmp1, 1) = stat_tmp2
stat_p(stat_tmp1, 2) = stat_tmp2 + 1
stat_p(stat_tmp1, 3) = stat_tmp2 + 2
```

```
        stat_p(stat_tmp1, 4) = stat_tmp2 + 3
```

After making this changes we were able to run successfully the code. We got the following error:

```
bc_switch_type[0:bc_switch_type.shape[0]:n_beta + n_radiation] = beta
ValueError: could not broadcast input array from shape (10) into shape (20
ValueError: output/normal_velocity/beta of value None should not be None.output/
normal_velocity/beta was not found in the hdf5 file at its location output/normal_velocity/beta
```

This error only happens when the number of wave frequencies is different than the number of wave directions.

The cause of this error is mainly that the preprocessor was failing due to an array shape issue and then the solver is run anyway causing it to fail also due to missing value not set by the preprocessor.

To solve the shape error (bc_switch_type[0:bc_switch_type.shape[0]:n_beta + n_radiation] = beta), we looked back at the original fortran code which was doing this:

```
    OPEN(11,FILE=ID%ID(1:ID%IID)//'/Normalvelocities.dat')
    WRITE(11,*) (Nbeta+Nradiation)*Nw
    WRITE(11,*) ((w(i),j=1,Nbeta+Nradiation),i=1,Nw)
    DO i=1,Nw
        WRITE(11,*) (/ (/(beta(j),j=1,Nbeta)/),(-1.,j=1,Nradiation) /)
 ENDDO
```

This means that we should set the first n_beta values of bc_switch_type to the array beta, then the next n_radiation values should be -1. We continue to do this process n_w times until the end of array bc_switch_type.

This is corrected using the following code:

```
bc_switch_type = -np.ones(n_problems, dtype='f') # Set the whole array to -1# Set the first
n_beta values to beta, skip the next n_radiation and so on until the end of the array
    for i in xrange(0, n_problems, n_beta + n_radiation):
        bc_switch_type[i:i + n_beta] = beta
```

Also we make sure the solver is not run in case of error in the preprocessor

These solved all the bugs and we were able to test for any value successfully.

Note that in theory any value is possible but we are limited by the available memory.

Testing for 360 and 500 respectively took more than 2 GB and run for 1h in our computer.

2- Verification

The simplest way to test is by using Linux:

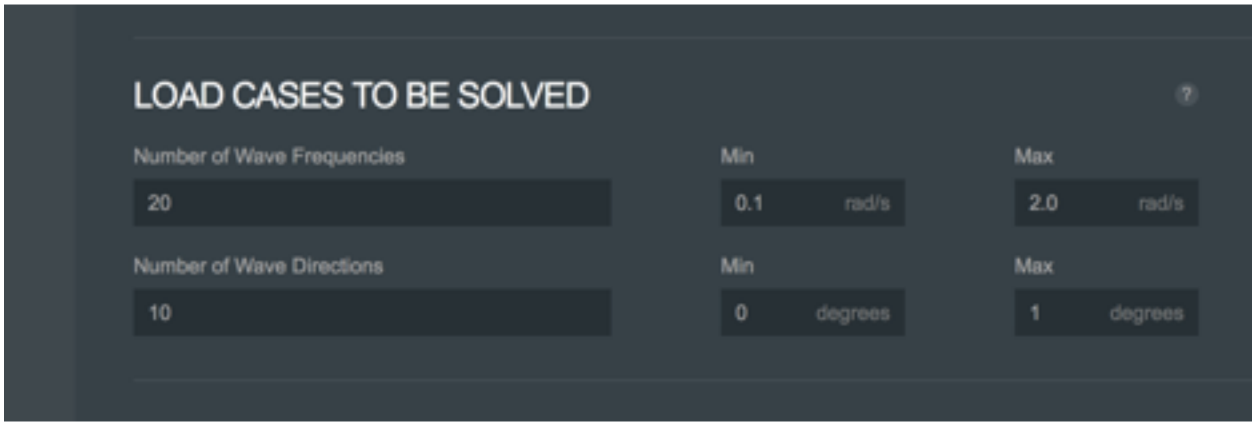
- Clone the git repository https://github.com/cloudspokes/OpenWARP and switch to branch 30053573_crash_fix
- Apply the patch file from our submission to get the new code using **git apply --whitespace=nowarn patch_file.patch** by replacing patch_file.patch with the location of the patch file.
- Download the linux installer from https://github.com/cloudspokes/OpenWARP/tree/master/installers
- Extract it and update the installer files with the file of the new code (the source directory after applying the patch)
- Run the file installer.sh with sudo
- Start the application

For OSX or Windows user, note that we added 2 new requirements for Python. So it is important to install them as explained in the deployment guide.

Now run the simulation by uploading the file Rotated_Cube.dat downloadable from http://apps.topcoder.com/forums/?module=GetAttachment&attID=129230

Then try by setting the number of wave frequencies to 10 and the number of wave directions to 10. Change their min and max value from 0.1 to 20 and from 0 to 200 respectively. You can pick other values too. Check that it works. You should see no error and a message with “INFO Problem” to be sure it is correctly run

Then try setting the number of wave directions to 20 (It is important the number of wave directions number and wave frequencies are different). See screenshots for example:



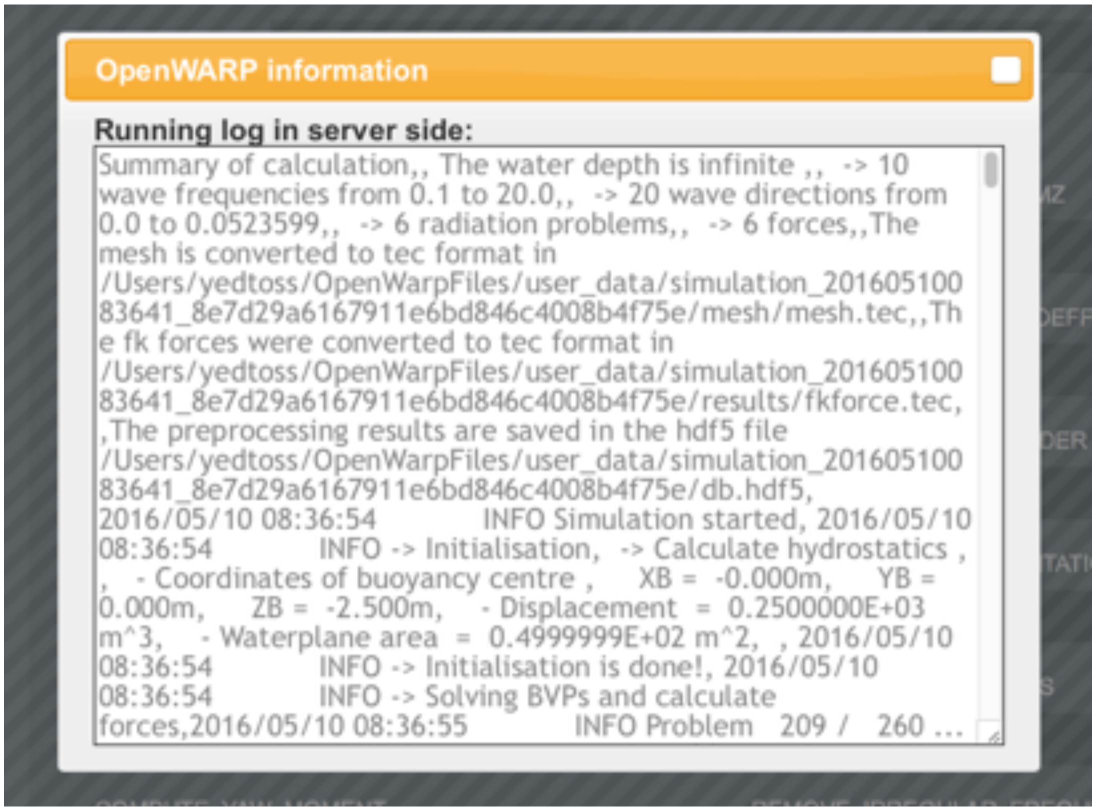
LOAD CASES TO BE SOLVED

Number of Wave Frequencies	Min	Max
20	0.1 rad/s	2.0 rad/s
Number of Wave Directions	Min	Max
10	0 degrees	1 degrees

Then try other values up to 150 for the wave frequencies and 300 for the wave directions and check that it works too. (This will take a lot of time (30 min in our computer) and more than 1GB)

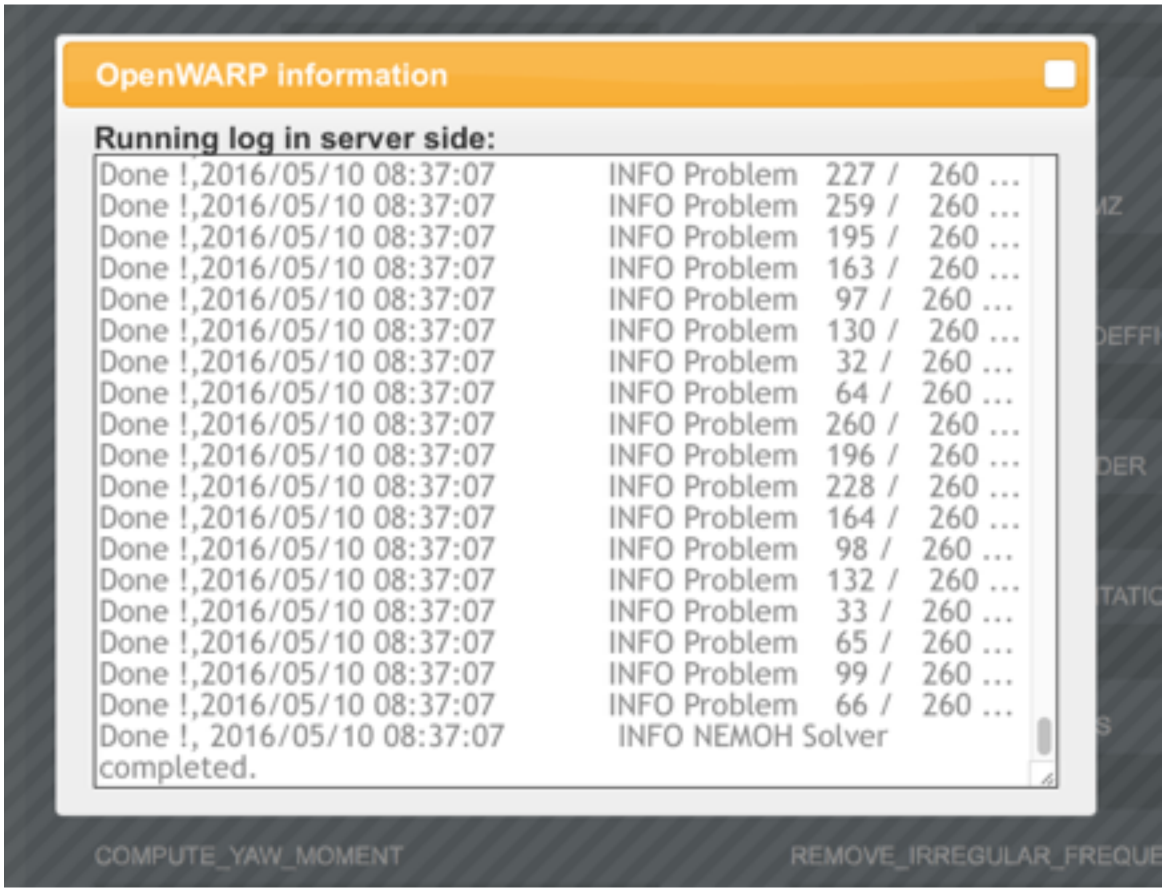
Finally try bigger values.

We included a log of what we did in the submission (It includes the error message as well as the success one). And here are two screenshots of a successful execution:



Running log in server side:

```
Summary of calculation,, The water depth is infinite ,,  -> 10
wave frequencies from 0.1 to 20.0,,  -> 20 wave directions from
0.0 to 0.0523599,,  -> 6 radiation problems,,  -> 6 forces,,The
mesh is converted to tec format in
/Users/yedtoss/OpenWarpFiles/user_data/simulation_201605100
83641_8e7d29a6167911e6bd846c4008b4f75e/mesh/mesh.tec,,Th
e fk forces were converted to tec format in
/Users/yedtoss/OpenWarpFiles/user_data/simulation_201605100
83641_8e7d29a6167911e6bd846c4008b4f75e/results/fkforce.tec,
,The preprocessing results are saved in the hdf5 file
/Users/yedtoss/OpenWarpFiles/user_data/simulation_201605100
83641_8e7d29a6167911e6bd846c4008b4f75e/db.hdf5,
2016/05/10 08:36:54            INFO Simulation started, 2016/05/10
08:36:54            INFO -> Initialisation,  -> Calculate hydrostatics ,
,   - Coordinates of buoyancy centre ,      XB =  -0.000m,      YB =
0.000m,      ZB =  -2.500m,      - Displacement  =  0.2500000E+03
m^3,     - Waterplane area  =  0.4999999E+02 m^2,   , 2016/05/10
08:36:54            INFO -> Initialisation is done!, 2016/05/10
08:36:54            INFO -> Solving BVPs and calculate
forces,2016/05/10 08:36:55            INFO Problem   209 /    260 ...
```



Running log in server side:

```
Done !,2016/05/10 08:37:07            INFO Problem   227 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem   259 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem   195 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem   163 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem    97 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem   130 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem    32 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem    64 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem   260 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem   196 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem   228 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem   164 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem    98 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem   132 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem    33 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem    65 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem    99 /    260 ...
Done !,2016/05/10 08:37:07            INFO Problem    66 /    260 ...
Done !, 2016/05/10 08:37:07            INFO NEMOH Solver
completed.
```