HDF5 Input and Output Implementation for NEMOH Deployment Guide

Revision History

Author	Revision Number	Date
yedtoss	1.0	21/10/2014

Deployment Instructions

1. Organization of Submission

Nemoh/ Contains the modified source of Nemoh Fortran software
docs/ Contains this deployment guide
tests/ Contains Original Nemoh Make file and the tools compare.py used to test the results
NemohPython/ Contains the python code
README.txt note about testing previous old fortran code

2. Application Setup

Linux

- GCC with Gfortran >= 4.8

- BLAS
- LAPACK
- 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 >= 4.8 http://www.cmake.org/cmake/resources/software.html
- Anaconda (with Python 2.7) >= 2.1.0 http://continuum.io/downloads

Windows

- 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 >= 4.8 http://www.cmake.org/cmake/resources/software.html
- Anaconda (with Python 2.7) >= 2.1.0 http://continuum.io/downloads

MAC OS X

- Brew 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 >= 4.8 installed by brew
- Anaconda (with Python 2.7) >= 2.1.0 http://continuum.io/downloads

Variables

Let's call:
$NEMOH_FORTRAN the directory Nemoh/ in the root of the submission directory
$NEMOH_PYTHON the directory NemohPython/
$FORTRAN_BUILD the build directory for the FORTRAN version of Nemoh
$MINGW_ROOT the directory where MINGW will be installed

3. Configuration

The python code can be configured using the file **$NEMOH_PYTHON/nemoh/settings.py**

Property	Definition	Example
HDF5_FILE	The path to the hdf5 file where to save and load the results and input. Required	'db.hdf5' No need to change
NEMOH_CALCULATIONS_FILE	The old nemoh calculation file. Not required but it is needed to automatically convert the old nemoh.cal file to hdf5 storage	'nemoh.cal' No need to change but then make sure you have the nemoh calculation file nemoh.cal in your working directory. Also make sure the path to the mesh file references in nemoh.cal exists. For exemple if in the nemoh.cal file you have 'Cylinder.dat', then you should have the Cylinder.dat file in your current directory
NEMOH_INPUT_FILE	Same as above but applied to the nemoh input file	'input.txt' No need to change but then make sure you have the nemoh input file input.txt in your working directory
NEMOH_INT	Represents the integer type to use when performing	'i'

Property	Definition	Example
	computations. It should be a valid numpy integer type. See http://docs.scipy.org/doc/numpy/reference/arrays.scalars.html#arrays-scalars-built-in for possible values	
NEMOH_FLOAT	Represents the float type to use when performing computations. It should be a valid numpy float type. See http://docs.scipy.org/doc/numpy/reference/arrays.scalars.html#arrays-scalars-built-in for possible values	'f'
NEMOH_COMPLEX	Represents the complex type to use when performing computations. It should be a valid numpy complex type. See http://docs.scipy.org/doc/numpy/reference/arrays.scalars.html#arrays-scalars-built-in for possible values	'F'
MESH_TEC_FILE	The path to the file where to save the mesh tec file	No need to change. The default value is fine.
FK_FORCE_TEC_FILE	The path to the froudkrylov force data in tec format	No need to change. The default value is fine.
RADIATION_COEFFICIENTS_TEC_FILE	The path to the file where to save the added mass and damping forces for the radiation problems in tec format.	No need to change. The default value is fine.
DIFFRACTION_FORCE_TEC_FILE	The path to the file where to save the diffraction force for the diffraction problems in tec format.	No need to change. The default value is fine.
EXCITATION_FORCE_TEC_FILE	The path to the file where to save the the excitation force for the diffraction problems in tec format.	No need to change. The default value is fine.
IRF_TEC_FILE	The path to the file where to save the IRF tec file	No need to change. The default value is fine.
WAVE_FIELD_TEC_FILE	The path to the file where to save the wave field tec file	No need to change. The default value is fine.

4. HDF5 Structure

The hdf5 structure is mostly defined in $NEMOH_PYTHON/nemoh/structure.py
You can configure the names as well as the attributes of the datasets in this file

You can find an example of a fully generated hdf5 file in $NEMOH_PYTHON/test_files/cylinder/data.hdf5

It is composed of 3 groups: the group “input” which contains all the input which should be given to the program.

The group “output” which contains intermediate output generated by the preprocessor

The group “results” which contains the results generated by the solver

You can directly modify the input group to change the input to the program.
You should also note that the current preprocessor contains a helper to convert the old nemoh input to the new one. To use it configure NEMOH_CALCULATIONS_FILE and NEMOH_INPUT_FILE property. See the Configuration section.

You can see a screenshot of the hdf5 structure at the end of this section.

Explanation about the structure

- Input/ contains the input
- /input/calculations/ contains all the parameters that were previously in “Nemoh.cal” including the mesh files like “Cylinder.dat”

- /input/calculations/bodies contains all the information about the bodies. It should contain one group per body. For example if we have two body it will contain body1 and body2

- /input/calculations/bodies/body1/mesh contains the mesh file for the first body. Basically what is inside “cylinder.dat”

  The other dataset inside /input/calculations/bodies/body1 contains descriptive name to identify their meaning

- /input/calculations/environment contains all datasets related to the environment. Each dataset has a descriptive name to help identify it against the old “Nemoh.cal”

- /input/solver contains dataset used for post processing. Basically it contains what was in the file “input.txt”. Dataset inside have descriptive names

- /output/mesh/l10 contains information inside the old L10.dat generated file
- /output/mesh/l12 contains information inside the old L12.dat generated file

- /output/mesh/free_surface contains information inside old FreeSurface.dat

- /output/mesh/kochin contain the kochin filename

- /output/mesh/integration contains the old Integration.dat information
- /output/normal_velocity contains information inside old NormalVelocities.dat

Please read the comments in the file $NEMOH_PYTHON/nemoh/structure.py for more comments about the groups and datasets

Implementation note

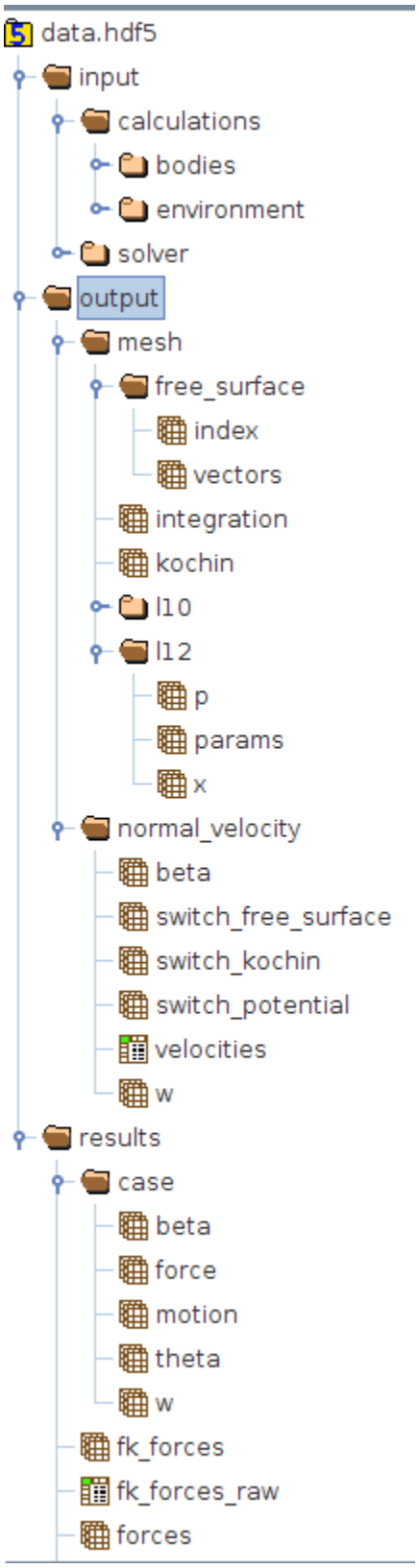
Basically what we have is a fortran library exposing the compute_nemoh routine.

Then we have define a cython module in solver_fortran.pyx to call it.

The file solver.py use the cython module to run the fortran subroutine

TEC FILES MAPPING

The current and previous tec files have similar names and are saved in a similar directory. They will only differ in the case or by the use of _


data.hdf5
input
calculations
bodies
environment
solver
output
mesh
free_surface
index
vectors
integration
kochin
l10
l12
p
params
x
normal_velocity
beta
switch_free_surface
switch_kochin
switch_potential
velocities
w
results
case
beta
force
motion
theta
w
fk_forces
fk_forces_raw
forces

5. Deployment Instructions

5.1. Generic Instructions

5.1.1. Install a Fortran and C/C++ compiler which support OpenMP (Currently gcc and ifort are supported)
5.1.2. Install Python 2.7
5.1.3. Install pip
5.1.4. Install CMake version greater or equal to 2.8
5.1.5. Install Blas and Lapack and make sure there are in the library search paths
5.1.6. Install hdf5 libraries version greater or equal to 1.8.11 and make sure there are in the library search path

5.1.7. Compile the Fortran version of Nemoh Solver:
  5.1.7.1. Create a build directory $FORTRAN_BUILD different from $NEMOH_FORTRAN
  5.1.7.2. Go to $FORTRAN_BUILD folder.
  5.1.7.3. Delete the file CMakeCache.txt and the directory CMakeFiles if it exists
  5.1.7.4. Run the command cmake
  5.1.7.5. Run make. The Nemoh library will be created

5.1.8. Compile the Nemoh python against the Nemoh fortran
  5.1.8.1. Go to $NEMOH_PYTHON
  5.1.8.2. Install the python module prerequisites pip install -r requirements.txt
  5.1.8.3. Make sure the $FORTRAN_BUILD is in the library search paths for compilation and for linking
  5.1.8.4. Run python setup.py build_ext --inplace

5.2. Linux Instructions with Ubuntu commands

(If you are not using Ubuntu, you should be able to use your distribution package manager to install equivalent commands)

5.2.1. Install GFortran and gcc by running: sudo apt-get install build-essential gfortran gcc

5.2.2. Install cmake by running sudo apt-get install cmake

5.2.3. Install Blas and Lapack and make sure there are in the library search paths by running sudo apt-get install liblapack-dev libblas-dev

5.2.4. Install python 2.7, hdf5 libraries, and the nemoh python module requirements. To do so, we just need to install Anaconda:
  5.2.4.1. Download Anaconda >= 2.1.0 from http://continuum.io/downloads .
    For linux 64 bits the direct link is (with no space) http://09c8d0b2229f813c1b93-c95ac804525aac4b6dba79b00b39d1d3.r79.cf1.rackcdn.com/Anaconda-2.1.0-Linux-x86_64.sh
  5.2.4.2. Install Anaconda by running bash Anaconda-2.1.0-Linux-x86_64.sh
  5.2.4.3. When prompted, accept to add it's path to your ~/.bashrc.
  5.2.4.4. Make sure the python version you are using is the one from Anaconda by logout then login or by running source ~/.bashrc
  5.2.4.5. If successful, when you run python --version you should see something like **Python**

2.7.8 :: Anaconda 2.1.0 (64-bit)

5.2.5. Compile the Nemoh Fortran
- 5.2.5.1. Create a build directory $FORTRAN_BUILD different from $NEMOH_FORTRAN
- 5.2.5.2. Go to $FORTRAN_BUILD folder by running cd $FORTRAN_BUILD .
- 5.2.5.3. Delete the file CMakeCache.txt and the directory CMakeFiles if it exists by running rm -rf CMakeCache.txt CMakeFiles/
- 5.2.5.4. Run cmake -DCMAKE_Fortran_COMPILER="gfortran" $NEMOH_FORTRAN to generate the Makefiles
- 5.2.5.5. Run make to build the library
- 5.2.5.6. The library libnemoh.so will be created

5.2.6. Compile the nemoh python against the nemoh fortran
- 5.2.6.1. Go to $NEMOH_PYTHON/nemoh
- 5.2.6.2. Make sure the $FORTRAN_BUILD is in the library search paths for compilation and for linking by running:
  export LD_LIBRARY_PATH=$FORTRAN_BUILD
  export LDFLAGS="-L$FORTRAN_BUILD"
- 5.2.6.3. Run python setup.py build_ext --inplace to build the python module in place

5.3. Windows Instructions

5.3.1. Install MinGW 4.8.1
- 5.3.1.1. You should install posix threads MinGW from http://sourceforge.net/projects/MinGWbuilds/files/host-windows/releases

  The 64 bits is located at 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

  The 32 bits is located at http://sourceforge.net/projects/MinGWbuilds/files/host-windows/releases/4.8.1/32-bit/threads-posix/sjlj/x32-4.8.1-release-posix-sjlj-rev5.7z/download

- 5.3.1.2. Download the binaries for your platform and extract it somewhere. By default the 64 bits get extracted in a directory named "MinGW64". Let's call this directory **$MINGW_ROOT**

- 5.3.1.3. Now add the full path to the folder **$MINGW_ROOT\bin** and **$MINGW_ROOT\lib** to your Path. Make sure those directories are at the left most (the beginning) of the Path. See **Setting Path on Windows** sub section for more information

5.3.2. Install CMake 2.8
- 5.3.2.1. Download and install cmake from http://www.cmake.org/files/v2.8/cmake-2.8.12.2-win32-x86.exe
- 5.3.2.2. Choose to add Cmake to the path for all users. By default Cmake doesn't put the path at the beginning.
  So, you need to make sure the cmake path is at the beginning of your path. It is by default " C:\Program Files (x86)\CMake 2.8\bin". See **Setting Path on Windows** sub section for more information

5.3.3. Install Lapack and Blas

Download and Install lapack and blas from http://icl.cs.utk.edu/lapack-for-windows/lapack/#libraries Choose the dll libraries for MinGW

For example, the 64 bits blas is http://icl.cs.utk.edu/lapack-for-windows/libraries/VisualStudio/3.5.0/Dynamic-MINGW/Win64/libblas.dll
And the 64 bits lapack is
http://icl.cs.utk.edu/lapack-for-windows/libraries/VisualStudio/3.5.0/Dynamic-MINGW/Win64/liblapack.dll

5.3.3.1. Copy them to **$MINGW_ROOT\lib** (the lib directory inside MinGW installation root)

5.3.4. Install Anaconda >= 2.1.0

5.3.4.1. Download and install Anaconda Windows version greater or equal to 2.1.0 from http://continuum.io/downloads . *The 64 bits Windows version is located at* http://09c8d0b2229f813c1b93-c95ac804525aac4b6dba79b00b39d1d3.r79.cf1.rackcdn.com/Anaconda-2.1.0-Windows-x86_64.exe

5.3.4.2. When installing accept adding anaconda to the path and using it's python version as the default python.
By default Anaconda would not put it's path to the beginning of the Windows Path.

You need to move the anaconda paths to the beginning of the Path list.
See **Setting Path on Windows** sub section for more information

For me the paths were C:\Users\yedtoss\Anaconda (the most important) and C:\Users\yedtoss\Anaconda\Scripts. You should replace C:\Users\yedtoss\Anaconda by the location where you install Anaconda

5.3.5. Compile the Nemoh Fortran

5.3.5.1. Start Powershell (Or windows cmd if you prefer it)
Make sure that all paths were correctly set. If not then you should close Powershell/Cmd, set the path and reopen it. Basically run python --version, cmake --version, gfortran --version and verify that they come from the one you just installed

5.3.5.2. Create a build directory $FORTRAN_BUILD different from $NEMOH_FORTRAN
5.3.5.3. Go to $FORTRAN_BUILD folder.
5.3.5.4. Delete the file CMakeCache.txt and the directory CMakeFiles if it exists *or better make sure* $FORTRAN_BUILD is empty
5.3.5.5. Run *cmake -DCMAKE_Fortran_COMPILER="gfortran" "*$NEMOH_FORTRAN**"** -G "MinGW Makefiles"

Please note the "" surrounding $NEMOH_FORTRAN. You need it even if the directory does not contain space. Make sure **"**$NEMOH_FORTRAN" is a full path to avoid any cmake bug

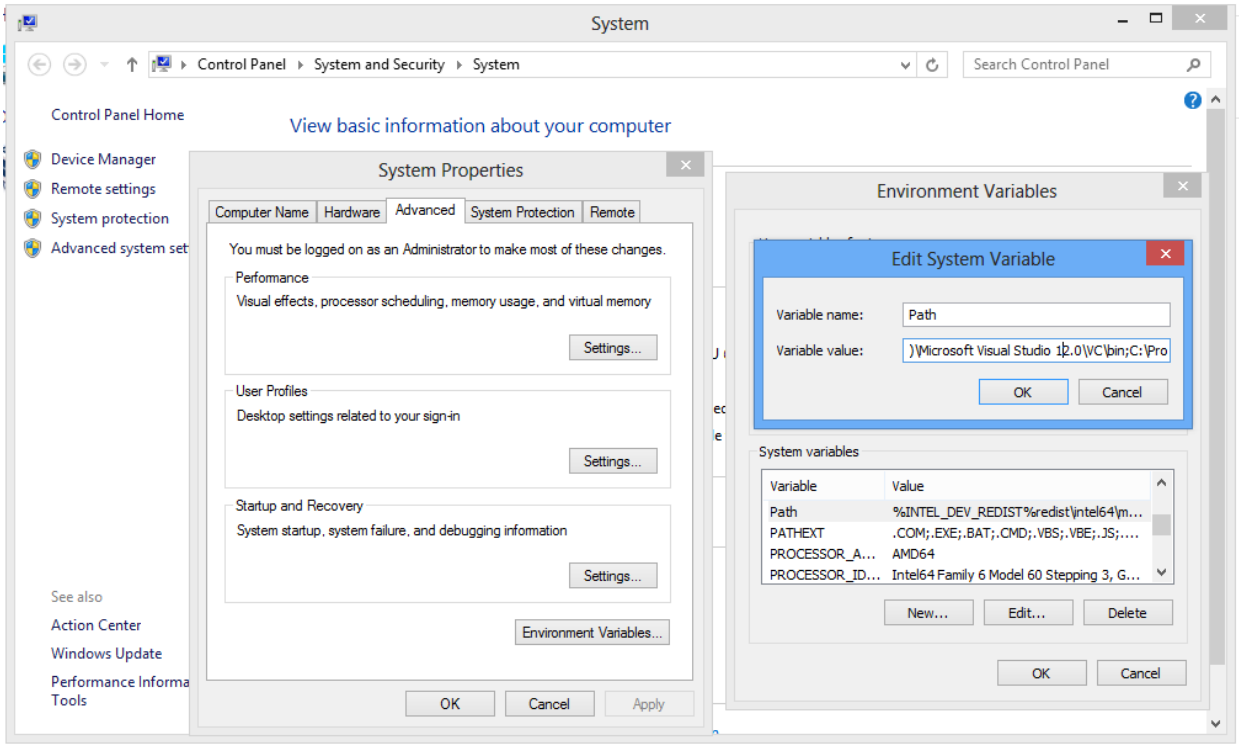
5.3.5.6. Run mingw32-make and you will get libnemoh.dll and libnemoh.dll.a
5.3.5.7. Copy both generated file to **$MINGW_ROOT\lib**

5.3.6. Compile the nemoh python against the nemoh fortran
5.3.6.1. Go to $NEMOH_PYTHON\nemoh
5.3.6.2. Run python setup.py build_ext --inplace

5.3.7. Setting Path on Windows

You need to go to Computer → Right click and choose Properties → Advanced System Settings → Advanced Tabs → Environment Variables → Look for path.

A screenshot



Note that when setting the path, you need to separate the different directories by ; Also make sure your new directory is not at the end of the list but at the beginning because Windows read the environment from left to right.

5.4. MAC OS X Instructions

We will use homebrew to install most software

5.4.1. Install brew: ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
(If you did not have XCode Command Line Tools, it will request it, install it and when done press enter on the terminal)

5.4.2. Run brew doctor

5.4.3. Install gcc and gfortran brew install gcc You can ignore any warning about multilib. It won't affect us

5.4.4. Install cmake brew install cmake

5.4.5. Download and Install anaconda from http://continuum.io/downloads
You can get the 64 bits version from http://09c8d0b2229f813c1b93-c95ac804525aac4b6dba79b00b39d1d3.r79.cf1.rackcdn.com/Anaconda-2.1.0-MacOSX-x86_64.pkg
We will export anaconda bin to the Path before compiling and using the nemoh python module

5.4.6. Compile the Nemoh Fortran

5.4.6.1. Create a build directory $FORTRAN_BUILD different from $NEMOH_FORTRAN

5.4.6.2. Go to $FORTRAN_BUILD folder cd $FORTRAN_BUILD .

5.4.6.3. Delete the file CMakeCache.txt and the directory CMakeFiles if it exists *rm -rf CMakeCache.txt CMakeFiles/*

5.4.6.4. Run cmake -DCMAKE_Fortran_COMPILER="gfortran" $NEMOH_FORTRAN

5.4.6.5. Generate nemoh fortran library by running make
The library libnemoh.dylib will be created
(If you get a warning about cmake policy ignore it)

5.4.7. Compile the nemoh python against the nemoh fortran

5.4.7.1. Go to $NEMOH_PYTHON/nemoh

5.4.7.2. Make sure you are using python from Anaconda
By running export PATH=/Users/tcs/anaconda/bin:$PATH
Replace /Users/tcs/anaconda/bin according to the location where anaconda was installed

5.4.7.3. Make sure the $FORTRAN_BUILD is in the library search paths for compilation and for linking by running
export LD_LIBRARY_PATH=$FORTRAN_BUILD
export LDFLAGS="-L$FORTRAN_BUILD*"*

5.4.7.4. Run python setup.py build_ext --inplace

Note that in the above process we did not explicitly install lapack or blas libraries. This is because it is implicitly installed with brew (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

Also note that the Intel Fortran Compiler (ifort) is fully supported on all three platforms. It comes bundled with lapack and blas so If you choose it, you won't need them.

Installing VMTK (Optional, you can skip)

VMTK is used by $NEMOH_PYTHON/nemoh/export_tec.py to export the generated .tec files to other

format. To install it for Mac, Windows or Linux follow instructions at
http://www.vmtk.org/documentation/installation.html
Note that unless you are using the .egg for Anaconda (Windows only), you should not use it with the python from Anaconda. More specifically, on Linux and Mac you have to use it with the built-in python when installing or using it.

6. Starting

You need to setup all environment variables as described in the deployment instructions.
You should also configure the application as described in the configuration section
Enter the directory $NEMOH_PYTHON/nemoh/
Run *python preprocessor.py* to run the preprocessor
Then run the solver with *python solver.py*
Finally run the post processor with *python postprocessor.py*

7. Verification

Setup your environment using the deployment instructions.
Then run the tools by following the Starting section

By default, the cylinder example has been configured. You can run other examples by modifying the configuration. You can find additional cases files in $NEMOH_FORTRAN/Verification folder
The hdf5 file db.hdf5 will be generated.

You can visualize it with HDFView from http://www.hdfgroup.org/products/java/release/download.html
It has a version for Windows, MAC and Linux

For example, to install the Ubuntu 64 bits:

- Download http://www.hdfgroup.org/ftp/HDF5/hdf-java/current/bin/HDFView-2.10.1-centos5-static64.tar.gz
- Extract it somewhere, then locate the file hdfview.sh inside it. For me it was located at HDFView-2.10.1-Linux/HDF_Group/HDFView/2.10.1/bin/hdfview.sh
- Edit the file hdfview.sh by setting INSTALLDIR to ■ ■
- Enter the bin directory
- Make sure you have java from Sun.
- Run bash hdfview.sh

Then open the hdf5 file

Correctness of the implementation

You need to run both the current python version and the old pure Fortran version against the same inputs.

You can verify the correctness by comparing the generated .tec files for both versions.
See section 4 (HDF5 Structure) to know the correct mapping to use.

To compare two files use the tool compare.py inside the tests/ directory

Use it like "python compare.py originalfile newfile"

Please see the old pure Fortran documentation for instructions about how to verify the solver is parallel, the gmres implementation and the speed of the solver.

Exporting to other format testing

The current code will not automatically export to other format. Instead you have to run:
From $NEMOH_PYTHON/nemoh python export_tec.py

This command will export all output tec files of the module to vtk file, stl file, ply file and "csv" file (space delimited). It will do this by converting the output .tec files to the target format using the vmtk utility http://www.vmtk.org . **Currently though this tool won't work against the output .tec files because those .tec files are wrongly formatted. This might be why they can't be visualize currently in paraview or tecplot (See http://apps.topcoder.com/forums/?module=Thread&threadID=833435&start=0)**

To prove that the tools do work we have included a correct sample .tec file located at
$NEMOH_PYTHON/test_files/cylinder/sample.tec

Run the tool like this python export_tec.py $NEMOH_PYTHON/test_files/cylinder/sample.tec
And it will generate .vtk, .pointdata, .stl and .ply files in the directory
$NEMOH_PYTHON/test_files/cylinder/

Note that you might need to remove python from Anaconda from the path to use it. See **Installing VMTK section.**

EFFECT OF NEMOH_INT, NEMOH_FLOAT and NEMOH_COMPLEX

In our testing, we found that the output of the python preprocessor and postprocessor can be affected by these variables. Currently they are set to match the precision of the previous Fortran code.
So they indicate the use of single precision for all calculations.

Note that single precision results (as done in previous Fortran code) is not at all precise especially when multiple intermediate computations are performed. The situation is noticeable for the FKForces when the values are small. We found that for single precision we can get a value of 0.3 whereas with better precision the true value is near 1e-10.

You can change those values to increase the precision of the python preprocessor and postprocessor. But it won't have any effect on the solver as the solver is ultimately performed by the fortran code.

8. Resource Contact List

Name	Resource Email
yedtoss	