Dipoles Implementation in NEMOH Deployment Guide

Revision History

Author	Revision Number	Date
yedtoss	1.0	10/01/2015

Deployment Instructions 3

1. Organization of Submission 3

2. Application Setup 3
   Linux 3
   Windows 3
   MAC OS X 3
   Variables 4

3. Configuration 4

4. Implementation details 6

5. Deployment Instructions 7
   5.1. Generic Instructions 7
   5.2. Linux Instructions with Ubuntu commands 7
   5.3. Windows Instructions 8
   5.4. MAC OS X Instructions 10

6. Starting 12

7. Verification 12

ACCURACY VERIFICATION 13

8. Limitations of the current dipoles implementation 13

9. References 14

10. Resource Contact List 14

Deployment Instructions

1. Organization of Submission

Nemoh/ Contains the modified source of Nemoh Fortran software
docs/ Contains this deployment guide
docs/GreenFunctionDerivativeODE.pdf Contains ODE for the derivative of the Green's function
docs/DipolesImplementationDetails.pdf Contains further details about the dipoles implementation
NemohPython/ Contains the modified Nemoh 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 >= 2.8 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 >= 2.8 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 >= 2.8 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**
Newly added properties are in red

Property	Definition	Example
USE_DIPOLES_IMPLEMENTATION	Whether or not to use the dipoles implementation	True
THIN_PANELS	A list containing the index of the panel which are thin dipoles. The index should be 0-based. Set the list to [-1] to indicate that all panels are thin dipoles.	[1, 2]
USE_HIGHER_ORDER	Whether or not to use higher order panel method.	True
NUM_PANEL_HIGHER_ORDER	The number of panel per patch in the higher order method. Read section 4 and 7 before attempting to increase this	1
B_SPLINE_ORDER	The order of the B-Spline for the potential in the higher order panel method. Read section 4 and 7 before attempting to increase this	1
USE_ODE_INFLUENCE_COEFF	Indicate whether or not to use the	True

Property	Definition	Example
ICIENTS	ode method to compute the influence coefficients	
GREEN_TABULATION_NUMX	Number of points in x direction of tabulated data	500
GREEN_TABULATION_NUMZ	Number of points in z direction of tabulated data	60
GREEN_TABULATION_SIMPSON_NPOINTS	Number of sub intervals used to approximate the Green's function integral using simpson rule	551
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 example 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 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	'i'
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	'f'

Property	Definition	Example
	for possible values	
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 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. Implementation details

The references are listed in section 9.

The main code for the dipoles implementation is written in "$NEMOH_FORTRAN/Solver/Core/SOLVE_BEM_DIPOLES_THIN.f90".

In the dipoles implementation, the body is assumed to be composed of two parts which can be empty. One part consists of conventional element S_s, the other is a surface consisting of thin 'dipoles' elements. To avoid singularity of the original potential (or source) formulation of the problem when the thickness of part of the body tends to zero, this method used an alternative integral equation. The alternative form is equation 15.46 and 15.47 of reference R2 (or equation 6.2 and 6.3 of reference R1). It is a pair of two simultaneous integral equations which is solved with a technique similar to the one describe in equation 15.3 of reference R2 or to Chapter 6 of reference R1. For further information please read Chapter 6 of reference R1 or section 15.7 of reference R2

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 they 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 $NEMOH_FORTRAN`
- 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 version 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 version 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 leftmost (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

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 than 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 files to **$MINGW_ROOT\lib**

5.3.6. Compile the nemoh python against the nemoh Fortran

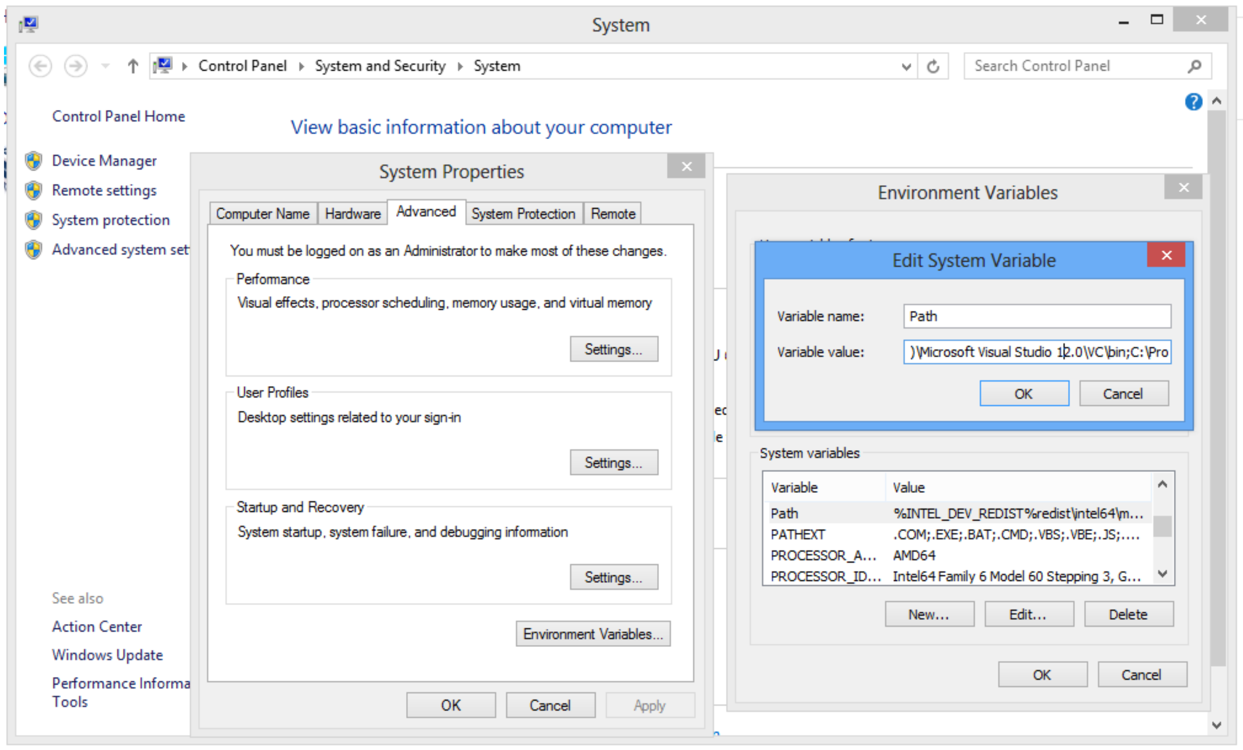
5.3.6.1. Go to $NEMOH_PYTHON\nemoh

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



System

Control Panel ▸ System and Security ▸ System

Search Control Panel

Control Panel Home

View basic information about your computer

Device Manager
Remote settings
System protection
Advanced system set

See also
Action Center
Windows Update
Performance Informa
Tools

System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...

User Profiles
Desktop settings related to your sign-in
Settings...

Startup and Recovery
System startup, system failure, and debugging information
Settings...

Environment Variables...

OK Cancel Apply

Environment Variables

Edit System Variable

Variable name: Path
Variable value:)\Microsoft Visual Studio 12.0\VC\bin;C:\Pro

OK Cancel

System variables

Variable	Value
Path	%INTEL_DEV_REDIST%redist\intel64\m...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	AMD64
PROCESSOR_ID...	Intel64 Family 6 Model 60 Stepping 3, G...

New... Edit... Delete

OK Cancel

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 reads the environment variables 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 formats. 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 example 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 to /tmp so that you have /tmp/HDFView-2.10.1-Linux, then locate the file hdfview.sh inside it. It should be located at /tmp/HDFView-2.10.1-Linux/HDF_Group/HDFView/2.10.1/bin/hdfview.sh
- Open the file hdfview.sh and set INSTALLDIR to ■■
- Enter the bin directory (/tmp/HDFView-2.10.1-Linux/HDF_Group/HDFView/2.10.1/bin)
- Make sure you have java from Sun, JRE is enough. You can use JRE 6 or JRE 7
- Run bash hdfview.sh

Then open the HDF5 file (Click File → Open)

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*

Ignore the warning of the type “Warning: normal vector of panel 14 points towards the x axis”. It is caused by the input mesh used.
The default example for test is $NEMOH_PYTHON/test_files/thin_body_dipoles

To run the dipoles implementation, set USE_DIPOLES_IMPLEMENTATION settings to True (the default) and set THIN_PANELS to the list of 'dipoles' panels or to [-1] to include all panels. To run previous lower order panel method (source formulation) set USE_DIPOLES_IMPLEMENTATION to False.

The lower order results are located in *$NEMOH_PYTHON/test_files/thin_body_dipoles/db_old.hdf5* and the dipoles implementation results in $NEMOH_PYTHON/test_files/thin_body_dipoles/db.hdf5

ACCURACY VERIFICATION

To test the accuracy of the dipoles implementation we can perform either one of the following tests:

We use a body containing thin mesh elements with a zero thickness. We run both the previous source formulation and the dipoles implementation against this test case. It is expected that the previous source formulation method will fail. Indeed the linear system to solve will be singular and the GAUSS solver will lead to singularities error. The dipoles implementation will successfully run.

We could also use a body containing thin mesh elements with thickness near zero (but not zero).
The body will be modeled with N panels
The previous could fail as before or could pass. If it passes, then we will take the same body and model it with 2N smaller panels. We will observe that the forces and potential will not converge and will be different.
Doing the same test with the dipoles implementation will converge

Finally it will be useful to use a body which has been already solved analytically (in the literature for example.) We could then compare the dipoles implementation and the previous source formulation against the analytical result.

We could not perform any of these tests because of the issue described here
http://apps.topcoder.com/forums/?module=Thread&threadID=844161&start=0

Instead we used a conventional body and we assumed that all its panel are thin dipoles ones.
Although this assumption is not correct, it shows that the dipoles implementation can be run successfully.
The results (dynamic forces) are different from the previous method because the integral equations we are solving is different in both cases. But the Froude Krylov Forces match between the dipoles implementation and the previous source formulation. This can be checked manually in the HDF5 files.

8. Limitations of the current dipoles implementation

- The current implementation does not compute the Kochin function and thus does not compute the drift forces, yaw moment and wave elevation (in postprocessing module). The Kochin function is not computed because the previous method to compute it uses the source strength which is not available in the implementation of the dipoles implementation. Other technique to compute the Kochin function should be performed.

- The dipoles implementation does not support the technique for symmetry in the body. You should describe the whole geometry of the body to use it. It should be possible to extend the dipoles implementation to support fast computation when there is symmetry.

To run your problem containing a symmetry in the mesh, disable the symmetry switch by setting the second number in the mesh file to 0 before running the preprocessor. Alternatively you can also set the second number in input/bodies/body1/mesh of the input HDF5 file to 0. You should do this for all bodies of your problem.
An error message will be given and program will stop if one attempts to use the new method with symmetry.

- The current dipoles implementation does not support computing the Green's function using an ODE as supported in the lower order panel method. If such an attempt is made, a warning will be shown and the method to compute the Green's function will fall back to the default (tabulation, series approximation and interpolation).

9. References

R1. **A Rankine panel method as a tool for the hydrodynamic design of complex marine vehicles.** http://dspace.mit.edu/bitstream/handle/1721.1/50364/39696248.pdf?sequence=1

R2. **Wamit User Manual** http://www.wamit.com/manualupdate/V70_manual.pdf

10. Resource Contact List

Name	Resource Email
yedtoss	