Refine Tabulation of the Green function of BEM code Nemoh Deployment Guide

Revision History

Author	Revision Number	Date
yedtoss	1.0	28/11/2014

Deployment Instructions

1. Organization of Submission

Nemoh/ Contains the modified source of Nemoh Fortran software
docs/ Contains this deployment guide
test/ Contains the test file simpson.py
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 >= 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**
Newly added properties are colored in red.

Property	Definition	Example
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 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 exemple if in the Nemoh.cal file you have 'Cylinder.dat', then you should have the Cylinder.dat file in your

Property	Definition	Example
		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 be used 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 be used 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 be used 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.

Property	Definition	Example
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. Implementation details

References

R1- http://lheea.ec-nantes.fr/lib/exe/fetch.php/emo/nemoh/delhommeau_papier052jh.pdf
R2 - http://lheea.ec-nantes.fr/lib/exe/fetch.php/emo/nemoh/thesedsgd010787.pdf
R3- http://lheea.ec-nantes.fr/lib/exe/fetch.php/emo/nemoh/aquaplus.pdf
R4- http://en.wikipedia.org/wiki/Simpson%27s_rule
R5- http://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.html

The main goal of this assembly is to refine the Green function tabulation to allow more accurate calculations of thin elements.

Let's first explain what Nemoh is about then explain how it is currently solving the problem and presents our solution.

The global goal of Nemoh is to solve the seakeeping problem. The seakeeping problem is the problem of bodies, floating (partially immersed) or immersed in a fluid of infinite or constant finite depth with or without forward speed submitted to sinusoidal waves. This is one of the main problems in hydrodynamics.

The mathematical formulation of the problem leads to the 3 dimensions radiation-diffraction problem of first order. Most code solving this problem uses the singularity model which can be described as follows:

The immersed part of the body is represented by thin quadrilateral shell elements. This representation is replaced by mathematical operators (the singularities). The resolution of the problem needs the constitution of system of linear equations whose coefficients (called influence coefficients) take into account the free surface (the surface not immersed in the fluid) using a Green function whose unknowns are the singularities. After solving those systems, we can then compute the forces on the body using the mechanic equations for movements.

The computation time of influence coefficient depends on the square of the number of shell elements and is cubic for solving the system of linear equations. For complex structures it is too much.
That is why many authors have proposed some methods to evaluate the Green function which allows solving the influence coefficients more quickly. The main idea is to approximate the Green function by a group of analytical functions each one defined in a part of the domain of variation of the Green function.

This is what Nemoh is currently doing. And the analytical form of the Green function can be found in R1

section 2.1 for the infinite depth problem and 2.4 for the finite depth problem. A more detailed and how those analytical functions are derived can be found in R2 section 2.2.

It can be seen that both the finite and infinite analytical functions used the same basic functions. See R1 section 2.3.1. Thus what Nemoh is doing is tabulating or approximating those basic functions and then used those approximations to compute the Green function.
The tabulation process is described in section 2.3.2 of R1.

Here is a short description of the current tabulation function. Note that the Green functions are integral whose solutions we don't know the analytical form.

To tabulate or approximate them, Nemoh computes the value of those integral for N points.
For each point the integral is computed using the simpson rule http://en.wikipedia.org/wiki/Simpson%27s_rule . To get the value of those integral at any point (or any point different from the N points chosen) the Lagrange interpolation formula (http://en.wikipedia.org/wiki/Lagrange_polynomial) is used.

Each point chosen is actually has a 2 dimensions one in X direction and one in Z direction.

The domain of validity of X direction (for the tabulation) is: 0 <= X <= 100
For Z is -1.5E-6 <= Z <=-16. Outside these domains we have other simpler formula for the Green functions because the Green functions tend to their asymptotic value. See R1 section 2.3.5 or R2 section 2.2.24 or 2.4 for more information.

So the tabulation should be restricted to the above domains.
Nemoh chooses 328 points in X directions and 46 points in Z directions giving a total of N = 46x328 points.
The precise value of each point for each pair I,J with 1<=I<=328 and 1<=J<=46 is given by the formula in section 2.3.2 R1.
The part of the Fortran code computing the (previous code) SUBROUTINE CREK(NPINTE) in Nemoh.f90
It computes the value of each point, then uses the SUBROUTINE VNS(NPINTE,AKZ,AKR,I,J) in Nemoh.f90 to compute the Simpson approximation of the integral (or Green function) at each point
According to the Fortran code the number of points used to approximate the integral is 251. See CALL CREK(251) in $NEMOH_FORTRAN/Solver/Core/INITIALIZATION.f90. All above files should be in previous code

Our Approach

To improve on the approximation, we need to increase both the number of points used in the tabulation and the number of sub intervals used for computing the integral by the Simpson rule. A change of the method used for computing the integral: The Simpson rule is not needed but an in-depth analysis might be needed.

When increasing the number of points in X directions and Z directions we need to keep in mind their domain and stay inside the domain. It is important that some points are more frequent than others and for that we keep the same method as Nemoh for 1<=I<=328 and 1<=J<=46.
For I>328 and J>46 we divide the respective variation domain in equidistant interval of length I or J respectively and take one point per such interval.

When increasing the number of sub intervals for the Simpson rule we need to keep in mind that the

resulting trapezoidal is not too small such that we can accurately compute its area.

All of these are implemented in the modified subroutines

SUBROUTINE COFINT(NPINTE,CQT,QQT) and SUBROUTINE CREK(NPINTE, n_tabulatedx, n_tabulatedz) in Nemoh.cal

ACCURACY

The accuracy of approximation of the integral is better than previous code if the configurable number of simpson interval is greater than 251.

This is because the bounded error depends on h = (b-a)/n (See R4 Composite Simpson's rule) with n the number of intervals. If n is greater then the error is smaller and the approximation is better

Also, the tabulation points that are considered by our modification include the previous points plus 0 or more points. This will always leads to more accuracy than previous approach because we are not changing the degree of Lagrange polynomial used in interpolating.
Thus the sentence in R5 which said: <<<When constructing interpolating polynomials, there is a tradeoff between having a better fit and having a smooth well-behaved fitting function. The more data points that are used in the interpolation, the higher the degree of the resulting polynomial, and therefore the greater oscillation it will exhibit between the data points. Therefore, a high-degree interpolation may be a poor predictor of the function between points, although the accuracy at the data points will be "perfect."<<< Does not affect us

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

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

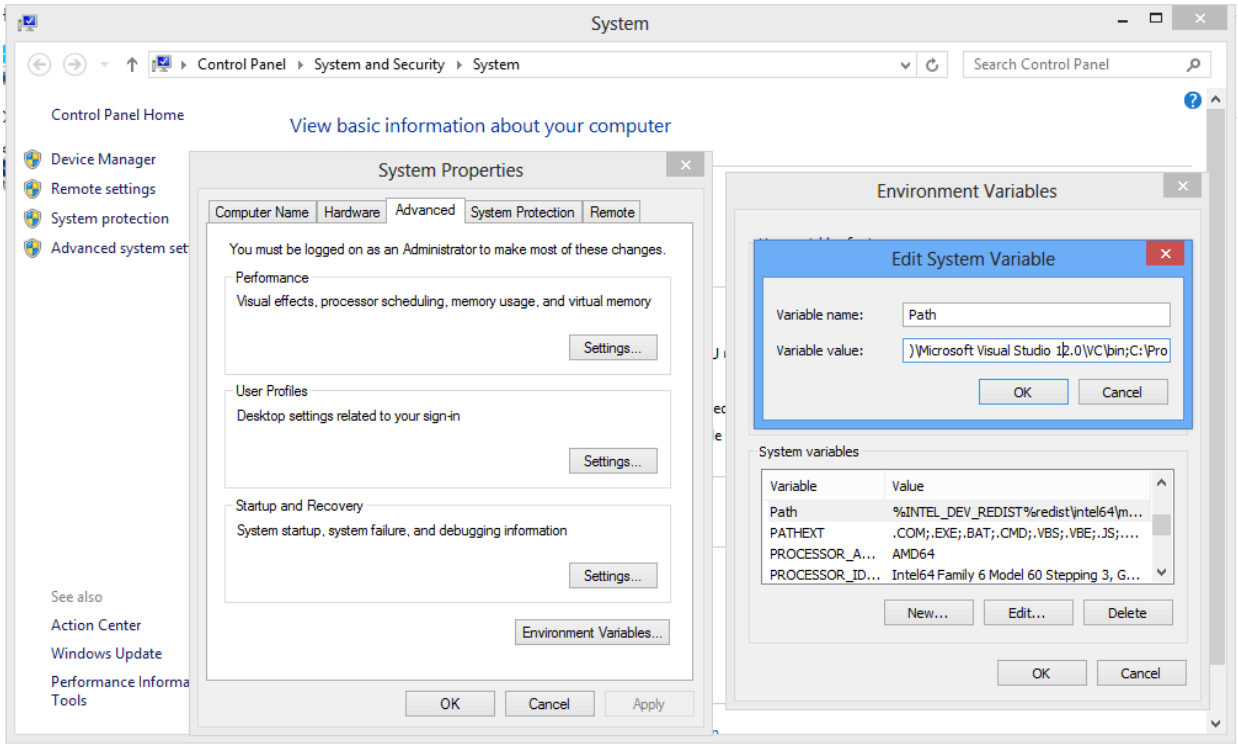
5.3.5.6. Run mingw32-make and you will get libnemoh.dll and libnemoh.dll.a
5.3.5.7. Copy both generated file to **$MINGW_ROOT\lib**

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 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 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

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*

ERRORS CHECKING

Modify $NEMOH_PYTHON/nemoh/settings.py

Change GREEN_TABULATION_SIMPSON_NPOINTS to 55
Then run python solver.py
We should get an error like
" Error: Please choose a number of sub intervals greater than 100 "

Change GREEN_TABULATION_SIMPSON_NPOINTS to 1 000 000

Then run python solver.py
We should get an error like
Error: Please choose a number of sub intervals less than 1 000 000
because all 1 000 000 surfaces will be thin.

IMPROVEMENT CHECKS

It is really difficult to test the original Nemoh code against the current one because we don't know the correct answer. In other words the correct integral value we are trying to approximate is not known. But it is well known in mathematics that with more sampling data, Simpson composite rule gives better results as demonstrated earlier.

For thin elements, we have an integral which is singular around the thin elements. So to approximate those integral it is important that we have enough sampling data. Most papers proposing some approximation other than Simpson or quadrature rule use these methods with enough points as the true solution.

To give a numerical proof of the method, we choose another singular integral where the answer is known and pick different step size for the simpson rule.

To execute it, go to the directory test/ at the root of submission and run *python simpson.py*

You will see something like

A similar example can be done for the Lagrange interpolation

8. Resource Contact List

Name	Resource Email
yedtoss	