Algorithmic Improvements to Nemoh

Topcoder

July 17, 2016

Contents

1 General approach to the seakeeping problem **4**

2 Higher-order panel method **5**

2.1 Improvement goal 5

2.2 Mathematical model 5

2.3 Implementation details and pseudocode 7

2.4 References 7

3 Dipole implementation for thin bodies **8**

3.1 Improvement goal 8

3.2 Mathematical model 8

3.3 Implementation details 9

3.4 References 9

4 Drift forces and QTF implementation **9**

4.1 Improvement goal 9

4.2 Mathematical model 10

4.3 Implementation details and pseudocode 11

4.4 References 12

5 Accelerating the calculation of influence coefficients **12**

5.1 Improvement goal 12

5.2 Mathematical model 12

5.3 Implementation details and pseudocode 14

5.4 References 14

6 Refining the tabulation of the Green's function **15**

6.1 Improvement goal 15

6.2 Mathematical model 15

6.3 Implementation details 16

6.4 References 17

7 Using the HDF5 file format **17**

7.1 Improvement goal 17

7.2 Architectural model 18

7.3 Implementation details 18

7.4 References 19

8 Test Cases **19**

8.1 Test 1: Converting GDF to OpenWarp Format 20

8.2 Test 2: Wamit Test Input and Output 20

8.3 Test 3: Testing Higher Order implementation in Openwarp . . . 21

8.4 Test 4: Testing the Dipoles/Thin Panels Implementation in Openwarp 21

8.5 Test 5: Testing the Irregular frequency removal in OpenWarp . . 22

8.6 Test 6: Testing the Mean Drift forces and Yaw moment implementation in Openwarp 22

8.7 Test 7: Computation of the green function using ordinary differential equation in OpenWarp 22

8.8 Test 8: Testing Green function Tabulation implementation 23

8.9 Test 9: 2Bodies 24

8.10 Test 10: Cylinder 24

8.11 Test 11: NonSymmetrical 24

1 General approach to the seakeeping problem

The overall goal of Nemoh is to solve the seakeeping problem, which is a central problem of hydrodynamics. In the seakeeping problem, we are concerned with bodies partially immersed (floating) or fully immersed in fluid of infinite depth or constant finite depth, with or without forward speed submitted to sinusoidal waves. The mathematical formulation of the problem leads to the three-dimensional radiation-diffraction problem of first order.

The seakeeping problem can be solved with the singularity model as follows. The immersed part of the body is represented by thin quadrilateral shell elements. These elements are replaced by mathematical operators called singularities. The problem is then described by a system of linear equations whose coefficients, called influence coefficients, take into account the free surface, which is the portion of the body surface that is not immersed in fluid. The system of linear equations is solved with a Green's function whose unknowns are the singularities. After solving the system of linear equations, we can compute the forces on the body using the mechanical equations of motion.

The computational costs of this approach depend on the number of shell elements. The time complexity of calculating the influence coefficients is $O(n^2)$ where n is the number of shell elements, and the time to solve the system of linear equations is $O(n^3)$.

The bodies may be subjected to sinusoidal waves in the fluid. We must then evaluate the loads and motions of the body in response to fluctuating hydrodynamic pressure and velocity. Nemoh solves this problem using the boundary element method (BEM).

Nemoh's algorithmic approach is the following.

1. The potential is represented either by a source distribution of unknown strength over the body surface, or by a Green's function in which the source strength is known and the dipole moment is equal to the unknown potential. These representations are called the source formulation and the potential formulation, respectively.
2. The body surface is approximated by a large number n of small quadrilateral panels in the lower-order panel method or by complex substructures in the higher-order panel method.
3. The source strength and dipole moment are assumed to be constant on each panel, giving a total of n unknowns.
4. In the source formulation, the normal derivative of the potential is evaluated at the centroid of each panel and set equal to the normal velocity

at that point, giving a total of n linear equations for the unknown source strengths. In the potential formulation, the potential is evaluated directly at the same points.

5. The system of equations is solved by standard methods of linear algebra.

6. With the resulting potentials, the pressure on each panel is evaluated and integrated to compute the required forces and moments.

2 Higher-order panel method

2.1 Improvement goal

In the higher-order panel method, we describe the potential with a non-linear approximation function instead of the piecewise constant function used in the lower-order panel method. When the bodies are described with a small number of complex subdivisions, the computation should be faster while yielding equal or better accuracy.

In the previous version of Nemoh, the velocity potential on each panel was assumed to be constant and equal to the value at the centroid of the panel. This assumption degraded the accuracy of the results. In the new implementation, the velocity potential at each point is described by a product of two B-splines.

2.2 Mathematical model

The surface S over which the integral equation is to be solved is decomposed into P patches. Each patch is expressed parametrically by a B-spline tension product expansion:

$$x^p(u,v) = \sum_{i=1}^{\bar{\mathcal{M}}_p} \sum_{j=1}^{\bar{\mathcal{N}}_p} x_{ij}^p \bar{U}_i^p(u) \bar{V}_j^p(v) \tag{1}$$

where $\bar{\mathcal{M}}_p = \bar{M}_p + \bar{k} - 1$ and $\bar{\mathcal{N}}_p = \bar{N}_p + \bar{k} - 1$.

$\bar{M}_p$ and $\bar{N}_p$ are the non-zero intervals between knots in the u and v parametric directions respectively.

$\bar{U}_p$ and $\bar{V}_p$ are the B-splines of order $\bar{k}$ in the parametric variables u and v respectively.

X_{ij}^p are the known vertices or coefficients. This means that the patches can be seen as a rectangular surface in the parametric space u, v.

On the rectangular parametric domain of each patch S_p, the potential is approximated with another B-spline tensor product expansion:

$$\phi(u,v) = \sum_{i=1}^{\mathcal{M}_p} \sum_{j=1}^{\mathcal{N}_p} \phi_{ij}^p U_i^p(u) V_j^p(v) \tag{2}$$

where $\mathcal{M}_p = M_p + k - 1$ and $\mathcal{N}_p = N_p + k - 1$.

M_p and N_p are the non-zero intervals between knots in the u and v parametric directions respectively.

U_p and V_p are the B-splines of order k in the parametric variables u and v respectively.

ϕ_{ij}^p are the known vertices or coefficients. This means that the patches can be seen as a rectangular surface in the parametric space u, v.

To obtain a discretization, we use the Galerkin method, leading to the equation

$$2\pi d_{ik}\phi_{jk} + \sum_{k=1}^{N} D_{ik}^H \phi_{jk} = S_i^H \tag{3}$$

where

$$d_{ik} = \int\int du_f \boldsymbol{U}_i(u_f)\boldsymbol{U}_k(u_f) \tag{4}$$

$$D_{ik}^H = \int\int du_f \boldsymbol{U}_i(u_f) \int\int du \boldsymbol{U}_k(u) \frac{\partial G(u,u_f)}{\partial n(u)} J(u) \tag{5}$$

$$S_i^H = \int\int du_f \boldsymbol{U}_i(u_f) \int\int du \boldsymbol{U}_k(u) \frac{\partial \phi_j}{\partial n}(u) G(u.u_f) J(u) \tag{6}$$

2.3 Implementation details and pseudocode

The system of linear equations is solved using a singular value decomposition. The integrals are solved numerically with a two-dimensional Gaussian quadrature.

The parametric space is derived as follows. Given a parametric coordinate pair u, v, we want to derive the corresponding Cartesian coordinates xx, yy, zz of a point in a panel.

Algorithm 1 Convert parametric coordinates to Cartesian coordinates

Given a parametric coordinate pair u, v,
calculate the Cartesian coordinates xx, yy, zz as folllows:
$X(M(i))$ is the x coordinate of the ith side of the panel
$Y(M(i))$ is the y coordinate
$Z(M(i))$ is the z coordinate
$N_1 = (1-u)(1-v)$
$N_2 = (1+u)(1-v)$
$N_3 = (1+u)(1+v)$
$N_4 = (1-u)(1+v)$
$xx = [N_1 \times X(M(1)) + N_2 \times X(M(2)) + N_3 \times X(M(3)) + N_4 \times X(M(4))]/4$
$yy = [N_1 \times Y(M(1)) + N_2 \times Y(M(2)) + N_3 \times Y(M(3)) + N_4 \times Y(M(4))]/4$
$zz = [N_1 \times Z(M(1)) + N_2 \times Z(M(2)) + N_3 \times Z(M(3)) + N_4 \times Z(M(4))]/4$

2.4 References

Maniar, H. D. *A Three-Dimensional Higher-Order Panel Method Based on B-Splines.* Massachusetts Institute of Technology, 1995.
http://dspace.mit.edu/bitstream/handle/1721.1/11127/34279481.pdf

WAMIT User Manual. WAMIT, Inc., 2013.
http://www.wamit.com/manualupdate/V70_manual.pdf

Danmeier, D. G. *A Higher-Order Panel Method for Large-Amplitude Simulations of Bodies in Waves.* Massachusetts Institute of Technology, 1999.
http://dspace.mit.edu/bitstream/handle/1721.1/9708/42663926.pdf

Liu, Y. *Fast Multipole Boundary Element Method.* Cambridge University Press, 2009.
https://books.google.bj/books?id=1p4SCnM5UIYC

Delhommeau, G. "Amélioration des performances des codes de calcul de diffraction-radiation au premier ordre." Journées de l'Hydrodynamique, 1989.
http://lheea.ec-nantes.fr/lib/exe/fetch.php/emo/nemoh/delhommeau_papier052jh.

pdf

Delhommeau, G. *Les Problèmes de Diffraction-Radiation et de Résistance de Vagues: Étude Théorique et Résolution Numérique par la Méthode des Singularités.* École Nationale Supérieure de Mécanique de Nantes, 1987.
http://lheea.ec-nantes.fr/lib/exe/fetch.php/emo/nemoh/thesedsgd010787.pdf

3 Dipole implementation for thin bodies

3.1 Improvement goal

The previous version of Nemoh used Green's theorem to represent the seakeeping problem but solved all integrals using a source formulation. This was problematic for bodies with thin elements, which occur in many marine applications. The discretization of integral equations using the standard formulation was computationally inefficient. In addition, it led to numerical instability due to the proximity of the Rankine panel collocation points on the two surfaces of the thin body.

In the new code, we treat the thin body as a single dipole sheet and reformulate the BEM to obtain an integral equation for the unknown strength of the dipole distribution.

3.2 Mathematical model

The body is considered to be infinitesimally thin and composed of two surfaces S_p- and S_p+ which occupy the same position but have opposite normal direction. The body surface is then considered as two parts: the thin body part S_d and the conventional part S_s.

The potential formulation of the integral equation leads to the following equations:

$$2\pi\phi(x) + \int\int_{S_s} \phi G_{n\epsilon} dS_{\epsilon +} \int\int_{S_d} \Delta\phi G_{n\epsilon} dS_\epsilon = \int\int_{S_s} \phi_{n\epsilon} G dS_\epsilon \qquad (7)$$

for $x \in S_s$, and

$$\int\int \phi G_{n_\epsilon n_x} dS_\epsilon + \int\int_{S_d} \Delta\phi G_{n_\epsilon n_x} dS_\epsilon = -4\pi\phi n_x + \int\int_{S_s} \phi_{n_\epsilon} G_{n_x} dS_\epsilon \quad (8)$$

for $x \in S_d$.

3.3 Implementation details

The integral is discretized and solved with a collocation method. In this method, we compute the integral for n known points, yielding a system of n linear equations. We solve the system of linear equations using Gaussian elimination.

The main algorithm loops through the panels twice before proceeding to solve the system of linear equations. In the first loop, we take the centroid (x, y, z) of each panel. In the second loop, we consider the coordinates of the quadrilateral of the corresponding panel.

We proceed by choosing n points corresponding to the discretized panels of the body. Each point corresponds to the center of gravity of a panel. The resulting equations form a linear system of the form $Ax = b$. To solve for A, we find the coefficients by Gaussian elimination with partial pivoting.

3.4 References

Mantzaris, D. A. *A Rankine Panel Method as a Tool for the Hydrodynamic Design of Complex Marine Vehicles.* Massachusetts Institute of Technology, 1999.
http://dspace.mit.edu/bitstream/handle/1721.1/50364/39696248.pdf

WAMIT User Manual. WAMIT, Inc., 2013.
http://www.wamit.com/manualupdate/V70_manual.pdf

Wikipedia: Collocation method
http://en.wikipedia.org/wiki/Collocation_method

4 Drift forces and QTF implementation

4.1 Improvement goal

We improve on the previous Nemoh code by computing second-order quantities which are important factors in the design of offshore structures. In the simula-

tion of slow drift motions, the mean force and moment are the most important quantities.

4.2 Mathematical model

We compute drift forces using the following equations.

$$\bar{F}_x = \frac{\rho v^2}{8\pi} \int_0^{2\pi} |H(\theta)|^2 \cos(\theta) d\theta + \frac{1}{2} \rho w A \cos(\beta) \Re \left(H(\pi + \beta) \right) \tag{9}$$

$$\bar{F}_y = \frac{\rho v^2}{8\pi} \int_0^{2\pi} |H(\theta)|^2 \sin(\theta) d\theta + \frac{1}{2} \rho w A \sin(\beta) \Re \left(H(\pi + \beta) \right) \tag{10}$$

$$\bar{M}_z = -\frac{\rho v}{8\pi} \Im \int_0^{2\pi} \bar{H}(\theta) \frac{\partial H}{\partial \theta} d\theta - \frac{1}{2} \frac{\rho w A}{v} \Im \left(\frac{\partial H(\pi + \beta)}{\partial \theta} \right) \tag{11}$$

In the above, $\bar{H}(\theta)$ is the complex conjugate of $H(\theta)$.

$$H(\theta) = \int \int_{S_B} \left(\frac{\partial}{\partial n} \phi - \phi \frac{\partial}{\partial n} \right) \exp^{v(2 - ix\cos(\theta) - iy\sin(\theta))} dS \tag{12}$$

The above equations use the far-field method of Maruo and work for fluid of infinite depth.

For fluid of finite depth, the far-field momentum formulas are used:

$$\bar{F}_x = \frac{-\rho g A^2}{m_0} \frac{C_g}{C} \left[\frac{1}{4\pi} Z^2 \int_0^{2\pi} |H_h(\theta)|^2 \cos(\theta) d\theta + 2\cos(\beta) \Im(H_h(\pi + \beta)) \right] \tag{13}$$

$$\bar{F}_y = \frac{-\rho g A^2}{m_0} \frac{C_g}{C} \left[\frac{1}{4\pi} Z^2 \int_0^{2\pi} |H_h(\theta)|^2 \sin(\theta) d\theta + 2\sin(\beta) \Im(H_h(\pi + \beta)) \right] \tag{14}$$

$$\bar{M}_z = \frac{-\rho g A^2}{m_0^2} \frac{C_g}{C} \left[\frac{1}{4\pi} Z^2 \Im \int_0^{2\pi} \bar{H}_h(\theta) \frac{\partial H_h}{\partial \theta} d\theta - \frac{1}{2} \Re \left(\frac{\partial H_h(\pi + \beta)}{\partial \theta} \right) \right] \tag{15}$$

In these equations, we have:

h = water depth

m_0 = wave number

ρ = mass of water

g = gravity

A = wave amplitude

The values Z and C_g/C are defined as follows.

$$Z = \frac{m_0^2 - v^2}{v^2 h - m_0^2 h - v} \cosh^2(m_0 h) \tag{16}$$

$$\frac{C_g}{C} = \frac{1}{2}\left(1 + \frac{2m_0 h}{\sinh(2m_0 h)}\right) \tag{17}$$

$H_h(\theta)$ is the finite-depth Kochin function:

$$H_h(\theta) = \int\int_{S_B} \left(\frac{\partial}{\partial n}\phi - \phi\frac{\partial}{\partial n}\right) \exp^{im_0(x\cos(\theta)+y\sin(\theta))} \frac{\cosh\left(m_0(Z+h)\right)}{\cosh(m_0 h)} dS \tag{18}$$

4.3 Implementation details and pseudocode

The integrals are computed with Romberg's trapezoidal method.

Algorithm 2 Romberg integration

1. Compute a trapezoid rule value with an initial step size h_0.
2. Cut the step size in half: $h_{i+1} = h_i/2$
3. Compute a new trapezoid rule value.
4. Go to step 2 and repeat until the difference of trapezoid rule values is less than the allowed error.

4.4 References

Kim, Y. *Computation of Higher-Order Hydrodynamic Forces on Ships and Offshore Structures in Waves.* Massachusetts Institute of Technology, 1999.
http://dspace.mit.edu/bitstream/handle/1721.1/79979/42664020.pdf

WAMIT User Manual. WAMIT, Inc., 2013.
http://www.wamit.com/manualupdate/V70_manual.pdf

5 Accelerating the calculation of influence coefficients

5.1 Improvement goal

In the seakeeping problem, influence coefficients are the coefficients of the system of linear equations that must be solved to obtain the potential. The main difficulty in computing the influence coefficients is computing the Green's function. With classical methods, the computational cost is cubic with respect to the number of panels. This leads to prohibitive running times when we are dealing with complex structures, especially in shallow water.

We improve on the traditional approach by dispensing with analytical expansions of Green's function. Instead, we approximate Green's function by interpolating values in four elementary functions of two parameters each. Calculations outside the parameters' domains of variation are performed with asymptotic functions.

5.2 Mathematical model

The Green's function is composed of two parts, the Rankine part and the wave part. The wave part is an integral function.

The wave part of the Green's function and its derivative constitute the solution of an exact differential equation. This technique only works for infinite-depth fluid. The following differential equations are used.

For the Green's function, we have

$$\frac{w^2}{4}G^{(2)} - w(w^2Z + \frac{3}{4})G^{(1)} + \left(w^4(r^2+Z^2) + w^2Z + 1\right)G = \frac{2(1+Zw^2)}{\sqrt{r^2+Z^2}} \quad (19)$$

with these initial conditions:

$$G(r, Z, 0) = \frac{2}{\sqrt{r^2 + Z^2}} - i(0) \tag{20}$$

$$\lim_{w \to 0} \left(\frac{\partial}{\partial w} G(r, Z, iw) \right) = 0 + i(0) \tag{21}$$

For the derivative G_R of the green function with respect to R, we have

$$\frac{w^2}{4} G_R^{(2)} - w(w^2 Z + \frac{9}{4}) G_R^{(1)} + \left(w^4 (r^2 + Z^2) + 3w^2 Z + 3 \right) G_R = \frac{-6r(1 + Zw^2)}{(r^2 + Z^2)^{\frac{3}{2}}} \tag{22}$$

with these initial conditions:

$$G(r, Z, 0) = \frac{-2r}{(r^2 + Z^2)^{\frac{3}{2}}} - i(0) \tag{23}$$

$$\lim_{w \to 0} \left(\frac{\partial}{\partial w} G(r, Z, iw) \right) = 0 + i(0) \tag{24}$$

For the derivative G_Z of the Green's function with respect to Z, we have

$$\frac{w^2}{4} G_R^{(2)} - w(w^2 Z + \frac{9}{4}) G_R^{(1)} + \left(w^4 (r^2 + Z^2) + 3w^2 Z + 4 \right) G_R = \frac{-8Z - w^2(4Z^2 - 2r^2)}{(r^2 + Z^2)^{\frac{3}{2}}} \tag{25}$$

with these initial conditions:

$$G(r, Z, 0) = \frac{-2Z}{(r^2 + Z^2)^{\frac{3}{2}}} - i(0) \tag{26}$$

$$\lim_{w \to 0} \left(\frac{\partial}{\partial w} G(r, Z, iw) \right) = 0 + i(0) \tag{27}$$

5.3 Implementation details and pseudocode

We tabulate the four functions with two logarithmic scales in Z. For X, we have a logarithmic scale followed by a linear scale.

Algorithm 3 Tabulation in Z

$N_2 \geq 46$, number of points in Z
for $J = 1, 20$ **do**
  $Z(J) = -10^{\frac{J}{5}-6}$
end for
for $J = 21, 46$ **do**
  $Z(J) = -10^{\frac{J}{8}-4.5}$
end for
for $J = 47, N_2$ **do**
  $Z(J) = \left(\frac{16}{N_2-46}(J-46) + 1.5 \times 10^{-6}\right)$
end for

Algorithm 4 Tabulation in X

$N_1 \geq 328$, number of points in X
$X(1) = 0$
for $I = 2, 31$ **do**
  $X(I) = 10^{\frac{I-1}{5}-6}$
end for
for $I = 32, 328$ **do**
  $X(I) = \frac{4}{3} + \frac{(I-32)}{3}$
end for
for $I = 329, N_1$ **do**
  $X(I) = \left(\frac{100}{N_1-328}(I-328)\right)$
end for

Note that the domain of variation for the tabulation of the variables X and Z is respectively $0 \leq X \leq 100$ and $-1.5E-6 \leq Z \leq -16$. For values outside this domain, a simple asymptotic formula is used.

To solve for the integral of the elementary function, Simpson's rule is used with a configurable number of steps.

The differential equations are solved using the Runge-Kutta method.

5.4 References

Newman, J. N. "Distributions of sources and normal dipoles over a quadrilateral panel." *Journal of Engineering Mathematics*, 1986.

http://link.springer.com/article/10.1007%2FBF00042771

Clément, A. H. "A second-order ordinary differential equation for the frequency-domain Green function." 1998.
http://www.iwwwfb.org/Abstracts/iwwwfb28/iwwwfb28_12.pdf

Clément, A. H. "An ordinary differential equation for the Green function of time-domain free-surface hydrodynamics." *Journal of Engineering Mathematics*, 1998.
http://link.springer.com/article/10.1023%2FA%3A1004376504969

Delhommeau, G. "Amélioration des performances des codes de calcul de diffraction-radiation au premier ordre." Journées de l'Hydrodynamique, 1989.
http://lheea.ec-nantes.fr/lib/exe/fetch.php/emo/nemoh/delhommeau_papier052jh.pdf

Delhommeau, G. *Les Problèmes de Diffraction-Radiation et de Résistance de Vagues: Étude Théorique et Résolution Numérique par la Méthode des Singularités.* École Nationale Supérieure de Mécanique de Nantes, 1987.
http://lheea.ec-nantes.fr/lib/exe/fetch.php/emo/nemoh/thesedsgd010787.pdf

6 Refining the tabulation of the Green's function

6.1 Improvement goal

After implementing a fast numerical approach to computing the Green's function, we want to increase the accuracy of the results. To do so, we increase the number of points used in the tabulation and the number of sub-intervals used to compute the integral.

6.2 Mathematical model

We express the Green's function and its derivative with four elementary functions:

$$
\begin{aligned}
D_1(Z,X) &= \Re \int_{-\pi/2}^{\pi/2} i\cos(\theta)\left[J(\zeta) - \frac{1}{\zeta}\right] d\theta \\
&= \Im \int_{-\pi/2}^{\pi/2} \cos(\theta)\left[J(\zeta) - \frac{1}{\zeta}\right] d\theta
\end{aligned}
\tag{28}
$$

$$
\begin{aligned}
D_2(Z,X) &= \Re \int_{-\pi/2}^{\pi/2} i\cos(\theta) e^{\zeta} d\theta \\
&= \Im \int_{-\pi/2}^{\pi/2} \cos(\theta) e^{\zeta} d\theta
\end{aligned}
\tag{29}
$$

$$
Z_1(Z,X) = \Re \int_{-\pi/2}^{\pi/2} \left[J(\zeta) - \frac{1}{\zeta}\right] d\theta \tag{30}
$$

$$
Z_2(Z,X) = \Re \int_{-\pi/2}^{\pi/2} e^{\zeta} d\theta \tag{31}
$$

with $\zeta = Z + iX\cos(\theta)$, $Z < 0$, $X \geq 0$.

In the domain of variation $\Im(\zeta) \geq 0$, the function $J(\zeta)$ can be computed directly from $E_1(\zeta)$ using $J(\zeta) = e^{\zeta}\left[E_1(\zeta) + i\pi\right]$.

For $X = 0$ and $Z \to 0^-$, the functions D_1 and Z_1 tend to infinity. With logarithmic behavior, D_2 and Z_2 tend to π.

$$
J(\zeta^0) = e^{\zeta}\left[E_1(\zeta^0) + i\pi\right] \quad \text{for} \quad \Im\zeta \geq 0 \tag{32}
$$

$$
J(\zeta^0) = e^{\zeta}\left[E_1(\zeta^0) - i\pi\right] \quad \text{for} \quad \Im\zeta < 0 \tag{33}
$$

To compute the Green's function at any point using the four elementary functions above, we must to able to compute D_1 and Z_1 for any input. We use Lagrangian interpolation to do so.

6.3 Implementation details

The set of points from which we interpolate are called the tabulated points. The more tabulated points we have, the better the approximation. Therefore,

we refine the calculation by adding to the tabulation a set of points obtained by dividing the domain of variation of each variable into n equal sub-intervals for a configurable value of n.

For each of the n points, we compute the value of the integral using Simpson's rule. To get the value of the integral at any point different from the n points, we use the Lagrange interpolation polynomial.

6.4 References

Delhommeau, G. "Amélioration des performances des codes de calcul de diffraction-radiation au premier ordre." Journées de l'Hydrodynamique, 1989.
http://lheea.ec-nantes.fr/lib/exe/fetch.php/emo/nemoh/delhommeau_papier052jh.pdf

Delhommeau, G. *Les Problèmes de Diffraction-Radiation et de Résistance de Vagues: Étude Théorique et Résolution Numérique par la Méthode des Singularités.* École Nationale Supérieure de Mécanique de Nantes, 1987.
http://lheea.ec-nantes.fr/lib/exe/fetch.php/emo/nemoh/thesedsgd010787.pdf

Delhommeau, G. "Numerical simulation of hydrodynamics: ships and offshore structures." École Centrale de Nantes, 1993
http://lheea.ec-nantes.fr/lib/exe/fetch.php/emo/nemoh/aquaplus.pdf

Wikipedia: Simpson's rule
http://en.wikipedia.org/wiki/Simpson%27s_rule

Mathworld: Lagrange Interpolating Polynomial
http://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.html

7 Using the HDF5 file format

7.1 Improvement goal

Nemoh originally used an assortment of plain text files for input and output. These files were numerous and written to various directories. To achieve clarity and maintainability, we have updated Nemoh so that it performs all input and output operations with the HDF5 file format. HDF5 files are organized hierarchically, somewhat like a file system. We took advantage of this feature to organize a file into groups, each one of which contains several data sets. The groups and data sets each have a list of attributes with ancillary information

such as descriptions of the contents.

7.2 Architectural model

In the new version of Nemoh, we organize an HDF5 file into three main groups: the input group, containing all input to the solver; the output group, containing all intermediate output of the preprocessor; and the result group, containing the final results of the solver and postprocessor.

Each data set in the HDF5 file contains a description of the data so that there is no need for external documentation.

7.3 Implementation details

The input and output functions of Nemoh have been delegated to a Python module that runs separately from the Fortran library, which is responsible for numerical computations. We wrote a Cython program to mediate between the Fortran library and the Python module.

In addition, the preprocessor for input data and the postprocessor for results were implemented in a Python module to ease future development.

In the new implementation, Nemoh files have an HDF5 structure described by the Python script `NemohPython/nemoh/structure.py`. For example, a Nemoh file has the following attributes for input data.

Path	Description
input/	top-level input group
input/solver/	input solver group
input/solver/USE_HIGHER_-ORDER	parameter indicating whether to use the higher-order panel method: 0 = no, 1 = yes
input/solver/REMOVE_IR-REGULAR_FREQUENCIES	parameter indicating whether to remove irregular frequencies: 0 = no, 1 = yes
input/solver/USE_DIPOLES_-IMPLEMENTATION	parameter indicating whether to use the dipole implementation: 0 = no, 1 = yes
input/solver/COMPUTE_-YAW_MOMENT	parameter indicating whether to compute the yaw moment: 0 = no, 1 = yes
input/solver/COMPUTE_-DRIFT_FORCES	parameter indicating whether to compute drift forces: 0 = no, 1 = yes
input/solver/THIN_PANELS	array of parameters indicating whether a given panel is conventional or dipole: 0 = conventional, 1 = dipole

7.4 References

Wikipedia: Hierarchical Data Format
http://en.wikipedia.org/wiki/Hierarchical_Data_Format

The HDF Group
http://www.hdfgroup.org/HDF5/

8 Test Cases

Although OpenWarp is a fork of Nemoh, it has features closer to Wamit than Nemoh. Under source /automated test folder, we re-use all the 25 test cases available in Wamit and compare the results obtained by OpenWarp against Wamit.

Following are the test cases that covers different features of OpenWarp against Wamit.

8.1 Test 1: Converting GDF to OpenWarp Format

Wamit recognizes geometry as GDF file only but Openwarp does not recognize it as input. During meshing, Openwarp takes input in STL, STEP, IGS format only and outputs it into DAT and GDF format. Later, Openwarp uses DAT format file only for simulation and postprocessing.

Thus, GDF files from Wamit can?t be used with Openwarp to do simulation, To achieve this we have made some changes in open source software named Bemio. We have changes done minor changes in Bemio?s source to convert GDF format to DAT format file

Location: https://github.com/lawsonro3/bemio/

Inside source/automated_test/Test1, GDFtoDAT.py is the script that makes use of automated_test/Test1/mesh.py to convert GDF files to DAT. This mesh.py file is copied from Bemio?s source and modified to create DAT file.

Optional: GDFMesh and RunNemoh source could also be modified to serve the purpose.

GDFMesh: http://lheea.ec-nantes.fr/lib/exe/fetch.php/emo/nemoh/gdfmesh.zip RunNemoh:http://lheea.ec-nantes.fr/lib/exe/fetch.php/emo/nemoh/runnemoh_v5.zip

Test1 tests whether Wamit's .gdf are compatible to use with Openwarp by converting it into corresponding .dat

8.2 Test 2: Wamit Test Input and Output

Openwarp produces output as a single db.hdf5 file. On the other hand Wamit takes input in GDF format and outputs are in OUT format. Also, it creates following 9 different types of file:

Pseudocode: 1. Read .OUT files using Bemio?s bemio.io.wamit 2. Convert it to .H5 using Bemio?s bemio.io.output 3. From Test 1, convert the .GDF file (Wamit?s input file) to corresponding .DAT file. 4. Update .json file to run with openwarp_cli.py, to create a db.HDF5 file. 5. Use H5DIFF tool to compare output from Step2 and Step4. 6. Calculate the differences between the objects and create a graph. Note: - Details of Wamit?s testruns could be found in v71_manual.pdf. All the tests are described and could be referred for details. Link:- http://www.wamit.com/manualupdate/V70_manual.pdf

Table 1: Wamit Output Type

Option	Description	Filename
1	Added Mass and damping coefficients	Frc.1
2	Exciting forces from Haskind relations	Frc.2
3	Exciting forces from diffraction potential	Frc.3
4	Motions of body (response amplitude operator)	Frc.4
5p	Hydrodynamic pressure on body surface	Frc.5p
5v	Fluid velocity vector body surface	Frc.(5vx,5vy,5vz)
6p	Pressure/free-surface elevation at field points	Frc.6p
6v	Fluid velocity vector at field points	Frc.(6vx,6vy,6vz)
7	Mean drift force and moment from control surface	Frc.7
8	Mean drift force and moment from momentum	Frc.8
9	Mean drift force and moment from pressure	Frc.9

8.3 Test 3: Testing Higher Order implementation in Openwarp

Testing of B_SPLINE_ORDER only in higher order not in lower order. Pseudocode:

1. Tun openwarp_cli.py with .dat file from Test 11 to Test 25 to create corresponding db.hdf5
2. Update .json file as use_higher_order = true and b_spline_order =1
3. Run openwarp_cli.py again with updated .json file from Test 11 to Test 25 to create new db.hdf5 files.
4. Use H5DIFF tool to compare the outputs from Step1 and Step 3.
5. Calculate the difference between the objects and create a graph.

8.4 Test 4: Testing the Dipoles/Thin Panels Implementation in Openwarp

Wamit?s test 01 and test 21 are suitable as they have dipoles enabled body in lower and higher order panels. Pseudocode:

1. Run openwarp_cli.py with .dat file for test 1 and test 21.
2. Update the .json file as use_dipoles_implemenation = true
3. Run openwarp_cli.py again with updated .json file for test1 and test21.

4. Use H5DIFF tool to compare the outputs from Step1 and Step3
5. Calculate the difference between the objects and create a graph.

8.5 Test 5: Testing the Irregular frequency removal in OpenWarp

Wamit?s test 02 and test 12 are suitable as with lower and higher order panel methods. Pseudocode:

1. Run openwarp_cli.py with .dat file for test2 and test 12.
2. Update the .json file as remove_irregular_frequencies=true
3. Run openwarp_cli.py again with updated .json file for test2 and test12.
4. Use H5DIFF tool to compare the outputs from Step1 and Step3
5. Calculate the difference between the objects and create a graph.

8.6 Test 6: Testing the Mean Drift forces and Yaw moment implementation in Openwarp

Any Wamit?s test can be used for this test. Horizontal drift forces and mean Yaw moment by momentum integration is getting tested here against Wamit?s output. Pseudocode:

1. Run openwarp_cli.py with .dat file for test1.
2. Update the .jso file as compute_drift_forces = true and compute_yaw_-moment=true
3. Run openwarp_cli.py again with update .json file for test1.
4. USE H5DIFF tool to compare the outputs from Step1 and Step3.
5. Calculate the difference between the objects and create a graph.

8.7 Test 7: Computation of the green function using ordinary differential equation in OpenWarp

Wamit does not support this ordinary differential equation, so we can make use of OpenWarp test files (e.g. flap-meshed). It test the difference in the output

with the green function, when ode is used to calculate the influence coefficients. Pseudocode:

1. Enable use_ode_influence coeffcients in one .json file
2. Run openwarp_cli.py to create db.hdf5 file.
3. Disable use_ode_influence_coefficients on other .json file
4. Run openwarp_cli.py to create db.hdf5 file.
5. Use H5DIFF tool to compare the outputs from Step 2 and Step 4.
6. Calculate the difference between the objects and create a graph.

8.8 Test 8: Testing Green function Tabulation implementation

This test is to see the difference in green function with changes in tabulated data. It also includes changing values in interval of Simpson?s rule.

- GREEN_TABULATION_NUMX - number of points in x direction of tabulated data
- GREEN_TABULATION_NUMZ - number of points in z direction of tabulated data
- GREEN_TABULATION_SIMPSON_NPOINTS - number of subintervals used to approximate the green function intervals.

Pseudocode:

1. Enable the above mentioned attributes of tabulation with a default value in .json file
2. Run openwarp_cli.py to create db.hdf5 file.
3. Change the above mentions attributes of tabulation as zero in .json file.
4. Run openwarp_cli.py to create db.hdf5 file.
5. Use H5DIFF tool to compare the outputs from Step 2 and Step 4.
6. Calculate the difference between the objects and create a graph.

8.9 Test 9: 2Bodies

Using Openwarp for simulation and postprocessing for 2Bodies geometry by creating a new .json file by taking values from Nemoh.cal and by importing cube.dat and rotated_cube.dat Pseudocode:

1. Update .json file with corresponding values in Nemoh.cal.
2. Import cube.dat and rotated_cube.dat
3. Execute simulation and post processing with common values and multiple simulation using openwarp_cli.

8.10 Test 10: Cylinder

Using Openwarp for simulation and postprocessing for Cylindrical geometry by creating a new .json file by taking values from Nemoh.cal and by importing cube.dat and rotated_cube.dat Pseudocode:

1. Update .json file with corresponding values in Nemoh.cal
2. Import Cylinder.dat
3. Execute simulation and post processing using openwarp_cli.

8.11 Test 11: NonSymmetrical

Using Openwarp for simulation and postprocessing for Nonsymmetrical geometry by creating a new .json file by taking values from Nemoh.cal and by importing nonsymmetrical.dat Pseudocode:

1. Update .json file with corresponding values in Nemoh.cal
2. Import NonSymmetrical.dat
3. Execute simulation and post processing using openwarp_cli