Report : Calculation of Added mass of a submerged body in Heave mode

Under the guidance of Guidance of
Mr. M Palaniappan
Scientist-F
National Institute of Ocean Technology(NIOT), Chennai

By: Shivam Verma
3rd year Undergraduate Student,
Ocean Engineering and Naval Architecture
Indian Institute of Technology(IIT), Kharagpur

Date : 22/10/20

Contents

1. Introduction

In case of freely floating body, or seakeeping problem, or for accelerating underwater vehicle, accurate calculation of added mass is very important to get the accurate prediction of the radiation force. Ideally, the radiation force may be calculated directly using Navier – Stokes (N-S) equation. However, potential theory based method is mostly used for the calculation of the added mass. In this project, lower order Rankine panel method is developed for the calculation of the heave added mass for a submerged rectangular box.

2. Mathematical Formulation

The general equation of continuity and equation of motion may be given as:

$$\frac{\partial \rho}{\partial t} + \Delta \cdot \left(\rho \vec{V}\right) = 0 \qquad (1)$$

$$\rho g_x - \frac{\partial \rho}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) = \rho \frac{Du}{Dt} \qquad (2)$$

$$\rho g_y - \frac{\partial \rho}{\partial y} + \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) = \rho \frac{Dv}{Dt} \qquad (3)$$

$$\rho g_z - \frac{\partial \rho}{\partial z} + \mu\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) = \rho \frac{Dw}{Dt} \qquad (4)$$

The above set of equations are called N-S Equations. The equation 1 represents the mass conservation and equation 2-4 represent momentum conservation and. The above set of equations can help in modelling any fluid flow problem if boundary conditions are rightly specified. However, these non-linear partial differential equations can only be solved using some numerical techniques as the known analytical techniques cannot handle these complex equations. The solutions are however greatly deviated from the actual flow phenomenon, due to cumulative errors of the applied numerical technique.

Here, however we are simplifying our problem by taking some assumptions coupled with some other equation, helps in arriving at equations which can we solved both analytically and numerically. The assumptions comprise the fluid is incompressible, inviscid, homogeneous and the flow is ir-rotational. Based on the above assumption, we can introduce a velocity potential ϕ such that the velocity of the fluid particle can be written as $V = \nabla \phi$. Also, if the fluid is assumed to be homogeneous, incompressible, then equation (1) takes the form

$$\nabla \cdot \vec{V} = 0 \qquad (5)$$

Substituting $V = \nabla \phi$ in equation (5), we get

$$\nabla^2 \phi = 0 \qquad (6)$$

In Cartesian co-ordinate system, the equation (6) can be written in the form:

$$\left(\frac{\partial^2 \varphi}{\partial x^2}+\frac{\partial^2 \varphi}{\partial y^2}\right)=0 \tag{7}$$

Equation (7) is called the Laplace Equation. Using the similar assumptions, the equation of motion given by (2-4) can be further modified as

$$p(\overline{X},t)=-\rho\left[\left(\frac{\partial \varphi}{\partial t}\right)+\frac{1}{2}(\nabla \varphi)^2+gz\right] \tag{8}$$

2.1 Gauss Divergence Theorem

The Gauss Divergence theorem relates the flux of a vector field through a closed surface to the divergence of the field in the volume enclosed.

$$\iint \vec{F}.\hat{n}\ \mathrm{dS} \quad = \quad \iiint \nabla \cdot (\vec{F})\ dV \tag{9}$$

2.2 Reynolds Transport Theorem

Reynolds transport theorem helps in transferring a control mass system into a control volume system.

$$\left.\frac{\partial N}{\partial t}\right)_{system}=\frac{\partial}{\partial t}\iiint_{cv} \eta\rho\ dV+\iint_{cs} \eta\rho\vec{V}\cdot d\vec{A} \tag{10}$$

$\left.\frac{\partial N}{\partial t}\right|_{system}$ is the rate of change of the system's extensive property N. For example, if $N=P$, we obtain the rate of change of momentum.

$\frac{\partial}{\partial t}\iiint_{cv} \eta\rho\ dV$ is the rate of change of amount of the property N in the control volume.

$\iint_{cs} \eta\rho\vec{V}\cdot d\vec{A}$ is the rate at which property N is exiting the surface of the control volume. The term $\eta\rho\vec{V}\cdot d\vec{A}$ computes the rate of mass transfer leaving across control surface area element $d\vec{A}$

For example:

$$\left.\frac{\partial P}{\partial t}\right)_{system}=\frac{\partial}{\partial t}\iiint_{cv} \rho\ dV+\iint_{cs} \rho\vec{V}\cdot d\vec{A} \tag{11}$$

3 Hydrodynamic Pressure Forces

One of the primary reasons for studying the fluid motion past a body is our desire to predict the forces and moments on the body due to dynamic pressure of the fluid. Thus, we wish to consider the six components of the force and moment vectors, which are represented by the integrals of the pressure over the body surface, or

$$F = \iint_{s_B} Pn \,\mathrm{d}S \tag{12}$$

$$M = \iint_{s_B} P(r \times n) \,\mathrm{d}S \tag{13}$$

4 Force on a Moving Body in an Unbounded Fluid

Under pure translation, we get equation (14) on expanding equation (12)

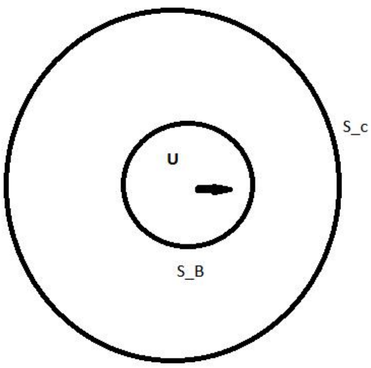
$$F_i = \rho \iint_{S_B} \left[\frac{\partial \varphi_j}{\partial t} + \frac{1}{2} \nabla \varphi_j \nabla \varphi_j \right] \cdot n_i \,\mathrm{d}S \tag{14}$$

Upon careful observation of φ_j, we can conclude that it tells the about the force in the ith direction, if the body is moving in the jth direction. Thus if the body is moving in the x direction and we are interested to know the force in the y direction, then equation (14) takes the form

$$F_2 = \rho \iint_{S_B} \left[\frac{\partial \varphi_1}{\partial t} + \frac{1}{2} \nabla \varphi_1 \nabla \varphi_1 \right] \cdot n_2 \,\mathrm{d}S \tag{15}$$

An alternative form of equation (15) can be written as

$$F_i = \rho \iint_{S_B} \left[\frac{\partial \varphi}{\partial t} n \right] dS \; + \rho \iint_{\mathrm{S_B}} \left(\frac{\partial \varphi}{\partial n} \right) \Delta \varphi n dS \tag{16}$$



Now, Applying Gauss divergent theorem and Raynolds transport theorem, we can get

$$\rho \frac{\partial}{\partial t} \iint_{S_B + S_C} \varphi \hat{n} \, dS = \rho \frac{\partial}{\partial t} \iiint_V \nabla \varphi \, dr$$

$$= \rho \iiint_V \nabla \left(\frac{\partial \varphi}{\partial t} \right) dV + \rho \iint_{S_B + S_C} \nabla \varphi (U \cdot \hat{n}) \, dS$$

$$= \rho \iint_{S_B + S_C} \left(\frac{\partial \varphi}{\partial t} \right) \cdot \hat{n} \, dS \; + \; \rho \iint_{S_B + S_C} \nabla \varphi (U \cdot \hat{n}) \, dS$$

now for $S_c, \vec{U} \cdot \hat{n} = 0$ and $\vec{U} \cdot \hat{n} = \frac{\partial \varphi}{\partial t}$, hence

$$\rho \frac{\partial}{\partial t} \iint_{S_B} \varphi_n \, dS = \rho \iint_{S_B} \left(\frac{\partial \varphi}{\partial t} \right) \cdot n \, dS + \iint_{S_B} \nabla \varphi \cdot \frac{\partial \varphi}{\partial n} \, dS$$

$$F = \rho \frac{\partial}{\partial t} \iint_{S_B} \varphi_n \, dS + \iint_{S_B} \frac{\partial \varphi}{\partial n} \nabla \varphi - \frac{(\nabla \varphi)^2}{2} n \, dS$$

Omitting the quadratic component gives the following expressions, we get:

$$F_i = \rho \frac{\partial}{\partial t} \iint_{S_B} \varphi_j n_i \, dS \tag{17}$$

$$M_i = \rho \frac{\partial}{\partial t} \iint_{S_B} \varphi_j (r \times n)_{i\text{-}3} \, dS \tag{18}$$

Suppose that the body has the translation velocity $\vec{U}$, then the velocity must satisfy the following boundry condition

$$\frac{\partial \varphi}{\partial n} = (\nabla \varphi) \cdot \hat{n} = \vec{U} \cdot \hat{n} \tag{20}$$

The boundary condition suggests that the total potential may be expressed as the sum

$$\varphi = u_i \bar{\varphi}_i \tag{21}$$

$$\frac{\partial \bar{\varphi}_i}{\partial n} = \hat{n}_i \tag{22}$$

$F_i = \rho \frac{d}{dt} u_j \iint_{S_B} \bar{\varphi}_j \hat{n}_i \, dS \Rightarrow F_i = \rho \dot{u}_j \iint_{S_B} \bar{\varphi}_j \frac{\partial \bar{\varphi}_i}{\partial n} dS$, which finally gives

$$F_i = m_{ij} \dot{u}_j \tag{23}$$

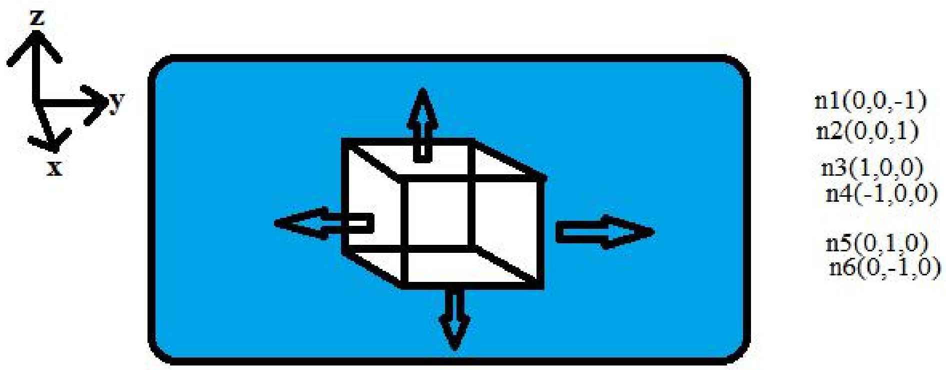
F_i : Force in the i^th direction due to the motion in the j^th direction.

$\bar{\varphi}_i$: Velocity potential due to unit velocity in the i^th direction.

5 Added Mass

In order to understand the added mass qualitatively we can intuitively understand that any body/vessel that moves in any fluid say water, will continuously try to push the fluid surrounded by the body. This extra work done by the body onto the fluid is called the added mass effect of the body. Added mass is a common issue because the body and surrounding fluid cannot occupy the same physical space simultaneously. For simplicity this can be modeled as some volume of fluid moving with the object, though in reality "all" the fluid will be accelerated, to various degrees. The added mass is a second-order tensor, relating the fluid acceleration vector to the resulting force vector on the body

There are many methods to quantify the added mass. Here, I have used **Panel method** to calculate the added mass.



To numerically compute the added mass, I have used the **Boundry Element Integral Equation method**. The procedure is :

1. Formulate the Boundary value Problem (for φ)
2. Find Proper Green's Function
3. Derive the appropriate Integral Equation
4. Discretize the boundary by small segments called panels
5. Distribute φ over the boundary
6. Apply initial/boundary conditions
7. Convert Integral Equation into algebraic Equation in the form [A]{ φ }={b}
8. Solve for φ.

5.1 Boundry Integral Equation

Solution of Boundry Integral Equation gives the distribution of phi(s) across all the faces of the submerged body $\overline{\varphi}(q)$ is the potential of source point Q, located on the face. This φ(q) is used to calculate the added mass on the body, as well as forces on the body. This φ(q) is called the **Radiation Potential**.

$$\alpha(P)\overline{\varphi}(P)=\iint_S\left[\overline{\varphi}(q)\frac{\partial G(p;q)}{\partial n_q}-G(p;q)\frac{\partial\overline{\varphi}(q)}{\partial n_q}\right]\mathrm{d}S \tag{24}$$

Where the surface S will be the combination of body, bottom and surface at infinity, however, for the present problem, the non trivial contribution on the integral is only coming from the body, therefore, the (24) may be re-written as:

$$\alpha(P)\overline{\varphi}(P)=\iint_{S_{body}}\left[\overline{\varphi}(q)\frac{\partial G(p;q)}{\partial n_q}-G(p;q)\frac{\partial\overline{\varphi}(q)}{\partial n_q}\right]\mathrm{d}S_{body} \tag{25}$$

$\overline{\varphi}(q)$ is the potential of the field point P. If P is on the surface, value of $\alpha(P)$ is 2π, if P is outside the body the value of $\alpha(P)$ is 4π and if P is inside the body then the value of $\alpha(P)$ is 0. Now define

$\varphi(P)$: Velocity potential

$$G(P,Q)=\frac{1}{R}=\frac{1}{\sqrt{(x-\xi)^2+(y-\eta)^2+(z-\zeta)^2}} \quad \text{where} \tag{26}$$

$P(x,y,z) \rightarrow$ Field point

$Q(\xi,\eta,\zeta) \rightarrow$ Source point

To get the computable form of the equation (25) we need to compute the value of $\frac{\partial G}{\partial n}$ as follows:

$$\frac{\partial G}{\partial n_q}=(\nabla G).(\hat{n}_q)=\left(\frac{\partial G}{\partial\xi}\hat{i}+\frac{\partial G}{\partial\eta}\hat{j}+\frac{\partial G}{\partial\zeta}\hat{k}\right)\cdot\left(n_\xi\hat{i}+n_\eta\hat{j}+n_\zeta\hat{k}\right)$$

Now

$R^2=(x-\xi)^2+(y-\eta)^2+(z-\zeta)^2$, which gives

$\frac{\partial R}{\partial\xi}=-\frac{(x-\xi)}{R},\frac{\partial R}{\partial\eta}=-\frac{(y-\eta)}{R},\frac{\partial R}{\partial\zeta}=-\frac{(z-\zeta)}{R}$, which gives

$\frac{\partial}{\partial\xi}\left(\frac{1}{R}\right)=-\frac{\partial}{\partial R}\left(\frac{1}{R}\right)\frac{\partial R}{\partial\xi}=\frac{(x-\xi)}{R^3}$, similarly

$\frac{\partial}{\partial\eta}\left(\frac{1}{R}\right)=\frac{(y-\eta)}{R^3}$ and $\frac{\partial}{\partial\zeta}\left(\frac{1}{R}\right)=\frac{(z-\zeta)}{R^3}$

Thus, finally we get

$$\frac{\partial G}{\partial n_q}=(\nabla G).(\hat{n}_q)=\left(\frac{\partial G}{\partial\xi}\hat{i}+\frac{\partial G}{\partial\eta}\hat{j}+\frac{\partial G}{\partial\zeta}\hat{k}\right)\cdot\left(n_\xi\hat{i}+n_\eta\hat{j}+n_\zeta\hat{k}\right)=\frac{\vec{R}.\hat{n}}{R^3} \tag{27}$$

The discretized form of equation (25) can be obtained as follows:

$$\alpha(P)\overline{\varphi}(P) = \iint\limits_{S_{body}} \left[\overline{\varphi}(q)\frac{\partial G(p;q)}{\partial n_q} - G(p;q)\frac{\partial \overline{\varphi}(q)}{\partial n_q} \right] \mathrm{d}S_{body}$$

$$\Rightarrow \alpha(P)\overline{\varphi}(P) = \sum_{i=0}^{n} \left[\left(\overline{\varphi}(q)\right)_i \left(\frac{\partial G(p;q)}{\partial n_q} \right)_i - (G(p;q))_i \left(\frac{\partial \overline{\varphi}(q)}{\partial n_q} \right)_i \right] dA_i$$

$$\Rightarrow \alpha(P)\overline{\varphi}(P) = \sum_{i=0}^{n} \left[\left(\overline{\varphi}(q)\right)_i \left(\left(\frac{\vec{R}_i}{|\vec{R}_i|^3} \right) \cdot (\hat{n}_i) \right) - \left(\frac{1}{\vec{R}_i} \right) (\hat{n}_i) \right] dA_i \tag{28}$$

ϕ1	ϕ2	ϕ3	ϕ4
ϕ5	ϕ6	ϕ7	ϕ8
ϕ9	ϕ10	ϕ11	ϕ12

Discretization of pannels and distribution of phi(s)

From equation (28) we can see that the field point is kept fixed and the source point moves.
In order to solve the equation (28) we assume that the field point is fixed on the 1st source point, i.e., φ_1 and hence the value of $\alpha(P)$ is 2π. And r is taken out for all the moving source points(φ_i) and fixed field points(φ_1). Then the 1st part of the RHS of equation (28) is assumed to be

$$\Rightarrow \alpha(P)\overline{\varphi}_1 = \sum_{\substack{i=1 \\ i\neq 1}}^{n} \left[\left(\overline{\varphi}\right)_i \left(\left(\frac{\vec{r}_i}{|r_i|^3} \right) \cdot (\hat{n}_i) \right) dA_i \;\; - (\frac{1}{r_i}) (\hat{n}_i)\, dA_i \right]$$

Assuming $\left(\left(\frac{\vec{r}_i}{|r_i|^3} \right) \cdot (\hat{n}_i) \right) = a_{1i}$ and $\sum_{\mathrm{i=0}}^{\mathrm{n}} (\frac{1}{r_i}) (\hat{n}_i) dA_i = b_1$, we can re write the above equation as

$$\alpha(P)\overline{\varphi}_1 + \overline{\varphi}_2 a_{12} + + \overline{\varphi}_3 a_{13} + ... + \overline{\varphi}_n a_{1n} = b_1 \tag{29}$$

Similarly, let $\overline{\varphi}$(P) be at φ_2

$$\Rightarrow \alpha(P)\overline{\varphi}_2 = \sum_{\substack{i=1 \\ i\neq 2}}^{n} \left[\left(\overline{\varphi}\right)_i \left(\left(\frac{\vec{r}_i}{|r_i|^3} \right) \cdot (\hat{n}_i) \right) dA_i \;\; - (\frac{1}{r_i}) (\hat{n}_i)\, dA_i \right]$$

Assuming $\left(\left(\frac{\vec{r}_i}{|r_i|^3}\right)\cdot(\hat{n}_i)\right)=a_{2i}$ and $\sum_{\substack{i=0\\i\neq 2}}^{n}(\frac{1}{r_i})(\hat{n}_i)dA_i=b_2$, we get

$$\overline{\varphi}_1 a_{21}+\alpha(P)\overline{\varphi}_2++\overline{\varphi}_3 a_{23}+.......+\overline{\varphi}_n a_{2\,n}=b_2 \tag{30}$$

Iterating on all the panels we end up with n number of equations, one corresponding to each panel. We arrange all the equations obtained as equations (29 and 39) into a matrix form. Thus the final matrix obtained is

$$\begin{bmatrix} \alpha(P) & a_{1,2} & a_{1,3} & \cdot & \cdot & \cdot & a_{1,n} \\ a_{2,1} & \alpha(P) & a_{2,3} & \cdot & \cdot & \cdot & a_{2,n} \\ a_{3,1} & a_{3,2} & \alpha(P) & \cdot & \cdot & \cdot & a_{3,n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{n,1} & a_{n,2} & a_{n,3} & \cdot & \cdot & \cdot & \alpha(P) \end{bmatrix} \begin{Bmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \cdot \\ \cdot \\ \varphi_{n-1} \\ \varphi_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ \cdot \\ \cdot \\ b_{n-1} \\ b_n \end{Bmatrix} \tag{31}$$

$$\Rightarrow [A]\{\varphi\}=\{B\}\Rightarrow\{\varphi\}=[A]^{-1}\{B\} \tag{32}$$

We solve for matrix $\{\varphi\}$. Then this $\{\varphi\}$ is used for evaluating the equation (24). We proceed as

$$m_{ij} \;=\rho\iint_{S_B}\overline{\varphi}_j\frac{\partial\overline{\varphi}_i}{\partial n}dS$$

Once we get the value of added mass, we can get the forces using equation (23).

5.2 Computation in Python and Output

Computing the matrix using the approach mentioned in equation (31).

```
import numpy as np
import sys

def matrix(npanels,face_coordn,face_norml,mode,face_ar):
    a_Fc=np.zeros((npanels,npanels))
    b_Fc=np.zeros(npanels)
    for i in range(0,npanels):
        r_fp=face_coordn[i]
        b_j=0
        for j in range(0,npanels):
            if j==i:
                a_Fc[i,j]=2*3.1414
                continue;
            r_sp=face_coordn[j]
            r=r_fp-r_sp
            n=face_norml[j]
            a_Fc[i,j]=(1*r[mode]*n[mode]*face_ar)/((sum(r**2)**1.5)*(sum(n**2)**0.5))
```

```
            b_j=b_j+((n[mode]*face_ar)/(sum(r**2)**0.5))
        b_Fc[i]=1*b_j
    return(a_Fc,b_Fc)

if __name__=="__main__":
#extracting the no. in between
    a=int(input('Enter Value of length : '))
    b=int(input('Enter value of breadth : '))
    c=int(input("Enter value of height : "))
    print("Density of water used : 1.025tonnes/m^3");
    print("....NOTE....")
    print("X is along front and back face")
    print("Y is along left and right face")
    print("Z is along bottom and top face")
    print("values of modes, heave=2, sway=1.surge=0");
    mode_k = int(input('Enter the mode : '))

    if mode_k >= 3:
        print("Wrong input of mode !, exiting")
        exit()

    x=np.linspace(0,1*a, (2*a+1))
    y=np.linspace(0,1*b, (2*b+1))
    z=np.linspace(0,1*c, (2*c+1))

    xx=np.array([x[i] for i in range(0,len(x)) if(i%2!=0)])
    yy=np.array([y[i] for i in range(0,len(y)) if(i%2!=0)])
    zz=np.array([z[i] for i in range(0,len(z)) if(i%2!=0)])
    ###panels centers coordinates storing
    Fc1=[]      #bottom
    Fc2=[]      #top
    Fc3=[]      #front
    Fc4=[]      #back
    Fc5=[]      #right
    Fc6=[]      #left

    for i in range(len(xx)):
        for j in range(len(yy)):
                Fc1.append(tuple((xx[i],yy[j],0*c)))    #bottom
                Fc2.append(tuple((xx[i],yy[j],1*c)))      #top
    for i in range(len(yy)):
        for j in range(len(zz)):
                Fc3.append(tuple((1*a,yy[i],zz[j])))    #front
                Fc4.append(tuple((0*a,yy[i],zz[j])))      #back
    for i in range(len(xx)):
        for j in range(len(zz)):
                Fc5.append(tuple((xx[i],1*b,zz[j])))    #right
                Fc6.append(tuple((xx[i],0*b,zz[j])))      #left

    Fc1=np.array(Fc1)           #bottom
```

```
Fc2=np.array(Fc2)           #top
Fc3=np.array(Fc3)           #front
Fc4=np.array(Fc4)          #back
Fc5=np.array(Fc5)          #right
Fc6=np.array(Fc6)          #left
##panels centers end

#panels normals making
n_Fc1=np.array([(0,0,-1) for i in range(0,len(Fc1))])     #bottom
n_Fc2=np.array([(0,0,1) for i in range(0,len(Fc2))]) #top
n_Fc3=np.array([(1,0,0) for i in range(0,len(Fc3))]) #front
n_Fc4=np.array([(-1,0,0) for i in range(0,len(Fc4))])#back
n_Fc5=np.array([(0,1,0) for i in range(0,len(Fc5))]) #right
n_Fc6=np.array([(0,-1,0) for i in range(0,len(Fc6))])#left
#panels normals end

mode=mode_k                  # 0 for surge,1 for sway, 2 for heave;
m=len(Fc1)          # no. of pannels m is same for all the faces
dA=1;

# for Face 1(bottom)
a_Fc1,b_Fc1=matrix(len(Fc1), Fc1, n_Fc1, mode, dA)
#for face 2(top)
a_Fc2,b_Fc2=matrix(len(Fc2), Fc2, n_Fc2, mode, dA)
#for face 3(front)
a_Fc3,b_Fc3=matrix(len(Fc3), Fc3, n_Fc3, mode, dA)
#for face 4(back)
a_Fc4,b_Fc4=matrix(len(Fc4), Fc4, n_Fc4, mode, dA)
#for face 5(right)
a_Fc5,b_Fc5=matrix(len(Fc5), Fc5, n_Fc5, mode, dA)
# for face 6(left)
a_Fc6,b_Fc6=matrix(len(Fc6), Fc6, n_Fc6, mode, dA)

# finding phi
phi_Fc1=np.dot(np.linalg.inv(a_Fc1),b_Fc1)
phi_Fc2=np.dot(np.linalg.inv(a_Fc2),b_Fc2)
phi_Fc3=np.dot(np.linalg.inv(a_Fc3),b_Fc3)
phi_Fc4=np.dot(np.linalg.inv(a_Fc4),b_Fc4)
phi_Fc5=np.dot(np.linalg.inv(a_Fc5),b_Fc5)
phi_Fc6=np.dot(np.linalg.inv(a_Fc6),b_Fc6)


print("Phi of bottom Face(face_1) : ",phi_Fc1)
print('\r')
print("Phi of Top Face(face_2) : ",phi_Fc2)
print('\r')
print("Phi of Front Face(face_3) : ",phi_Fc3)
print('\r')
print("Phi of Back Face(face_4) : ",phi_Fc4)
print('\r')
```

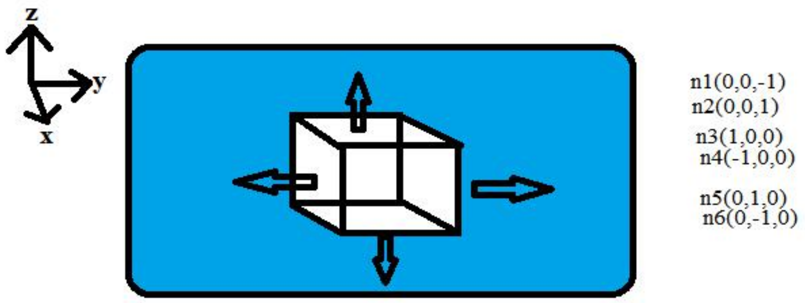
```
print("Phi of Right Face(face_5): ",phi_Fc5)
print('\r')
print("Phi of Left Face(face_6) : ",phi_Fc6)
print('\r')



print('Added Mass........')
m_1_1=0      #bottom face added mass
m_2_2=0      #top face added mass
m_3_3=0      #front
m_4_4=0      #back
m_5_5=0      #right
m_6_6=0          #left

rho=1.025

if (mode==2):                                    #heave (top-bottom)
    for i in range(len(phi_Fc1)):
        m_1_1=m_1_1+phi_Fc1[i]*(n_Fc1[i][mode])
    for i in range(len(phi_Fc2)):
        m_2_2=m_2_2+phi_Fc2[i]*(n_Fc2[i][mode])
    m_2_2=dA*rho*m_2_2
    m_1_1=dA*rho*m_1_1
    print('Heave Added mass : ',(m_1_1+m_2_2))
elif (mode==1):                                  #sway (left-right)
    for i in range(len(phi_Fc5)):
        m_5_5=m_5_5+phi_Fc5[i]*(n_Fc5[i][mode])
    for i in range(len(phi_Fc6)):
        m_6_6=m_6_6+phi_Fc6[i]*(n_Fc6[i][mode])
    m_6_6=dA*rho*m_6_6
    m_5_5=dA*rho*m_5_5
    print('Sway Added mass : ',(m_6_6+m_5_5))
elif (mode==0):                                  #surge (front-back)
    for i in range(len(phi_Fc3)):
        m_3_3=m_3_3+phi_Fc3[i]*(n_Fc3[i][mode])
    for i in range(len(phi_Fc4)):
        m_4_4=m_4_4+phi_Fc4[i]*(n_Fc4[i][mode])
    m_4_4=dA*rho*m_4_4
    m_3_3=dA*rho*m_3_3
    print('Surge Added mass : ',(m_3_3+m_4_4))
```



Output

Output : 1 [Heave added mass calculation]

Added Mass Calculation of a submerged rectangular body
Enter Value of length : 5
Enter value of breadth : 4
Enter value of height : 6
Density of water used : 1.025tonnes/m^3
....NOTE....
X is along front and back face
Y is along left and right face
Z is along bottom and top face
values of modes, heave=2, sway=1.surge=0
Enter the mode : 2
Phi of bottom Face(face_1) : [-1.26707112 -1.48201833 -1.48201833 -1.26707112 -1.51447765 -1.78486883
-1.78486883 -1.51447765 -1.58061116 -1.86522179 -1.86522179 -1.58061116
-1.51447765 -1.78486883 -1.78486883 -1.51447765 -1.26707112 -1.48201833
-1.48201833 -1.26707112]

Phi of Top Face(face_2) : [1.26707112 1.48201833 1.48201833 1.26707112 1.51447765 1.78486883
1.78486883 1.51447765 1.58061116 1.86522179 1.86522179 1.58061116
1.51447765 1.78486883 1.78486883 1.51447765 1.26707112 1.48201833
1.48201833 1.26707112]

Phi of Front Face(face_3) : [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Phi of Back Face(face_4) : [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Phi of Right Face(face_5): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0.]
Phi of Left Face(face_6) : [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0.]
Added Mass........
Heave Added mass : 63.72508966381723

Output : 2 [Sway added mass calculation]

```
Added Mass Calculation of a submerged rectangular body
Enter Value of length : 10
Enter value of breadth : 7
Enter value of height : 5
Density of water used : 1.025tonnes/m^3
....NOTE....
X is along front and back face
Y is along left and right face
Z is along bottom and top face
values of modes, heave=2, sway=1.surge=0
Enter the mode : 1
Phi of bottom Face(face_1) :    [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Phi of Top Face(face_2) :    [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Phi of Front Face(face_3) :    [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Phi of Back Face(face_4) :    [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Phi of Right Face(face_5):    [1.99802035 2.27247534 2.34818786 2.27247534 1.99802035 2.34438375
 2.691366     2.78744044 2.691366     2.34438375 2.53579153 2.91641729
 3.02394787 2.91641729 2.53579153 2.64492014 3.04127793 3.15447377
 3.04127793 2.64492014 2.69521513 3.09794507 3.213459     3.09794507
 2.69521513 2.69521513 3.09794507 3.213459     3.09794507 2.69521513
 2.64492014 3.04127793 3.15447377 3.04127793 2.64492014 2.53579153
 2.91641729 3.02394787 2.91641729 2.53579153 2.34438375 2.691366
 2.78744044 2.691366     2.34438375 1.99802035 2.27247534 2.34818786
 2.27247534 1.99802035]

Phi of Left Face(face_6) :    [-1.99802035 -2.27247534 -2.34818786 -2.27247534 -1.99802035 -2.34438375
 -2.691366     -2.78744044 -2.691366     -2.34438375 -2.53579153 -2.91641729
 -3.02394787 -2.91641729 -2.53579153 -2.64492014 -3.04127793 -3.15447377
 -3.04127793 -2.64492014 -2.69521513 -3.09794507 -3.213459     -3.09794507
 -2.69521513 -2.69521513 -3.09794507 -3.213459     -3.09794507 -2.69521513
 -2.64492014 -3.04127793 -3.15447377 -3.04127793 -2.64492014 -2.53579153
 -2.91641729 -3.02394787 -2.91641729 -2.53579153 -2.34438375 -2.691366
 -2.78744044 -2.691366     -2.34438375 -1.99802035 -2.27247534 -2.34818786
 -2.27247534 -1.99802035]

Added Mass........
Sway Added mass :    274.71284945186056
```

Output : 3 [Surge added mass calculation]

Added Mass Calculation of a submerged rectangular body
Enter Value of length : 8
Enter value of breadth : 8
Enter value of height : 8
Density of water used : 1.025tonnes/m^3
....NOTE....
X is along front and back face
Y is along left and right face
Z is along bottom and top face
values of modes, heave=2, sway=1.surge=0
Enter the mode : 0
Phi of bottom Face(face_1) : [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Phi of Top Face(face_2) : [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Phi of Front Face(face_3) : [2.36299746 2.71511943 2.89841583 2.98107006 2.98107006 2.89841583
2.71511943 2.36299746 2.71511943 3.15084733 3.37455871 3.47393365
3.47393365 3.37455871 3.15084733 2.71511943 2.89841583 3.37455871
3.62296877 3.73358398 3.73358398 3.62296877 3.37455871 2.89841583
2.98107006 3.47393365 3.73358398 3.84963092 3.84963092 3.73358398
3.47393365 2.98107006 2.98107006 3.47393365 3.73358398 3.84963092
3.84963092 3.73358398 3.47393365 2.98107006 2.89841583 3.37455871
3.62296877 3.73358398 3.73358398 3.62296877 3.37455871 2.89841583
2.71511943 3.15084733 3.37455871 3.47393365 3.47393365 3.37455871
3.15084733 2.71511943 2.36299746 2.71511943 2.89841583 2.98107006
2.98107006 2.89841583 2.71511943 2.36299746]

Phi of Back Face(face_4) : [-2.36299746 -2.71511943 -2.89841583 -2.98107006 -2.98107006
-2.89841583
-2.71511943 -2.36299746 -2.71511943 -3.15084733 -3.37455871 -3.47393365
-3.47393365 -3.37455871 -3.15084733 -2.71511943 -2.89841583 -3.37455871
-3.62296877 -3.73358398 -3.73358398 -3.62296877 -3.37455871 -2.89841583
-2.98107006 -3.47393365 -3.73358398 -3.84963092 -3.84963092 -3.73358398
-3.47393365 -2.98107006 -2.98107006 -3.47393365 -3.73358398 -3.84963092
-3.84963092 -3.73358398 -3.47393365 -2.98107006 -2.89841583 -3.37455871
-3.62296877 -3.73358398 -3.73358398 -3.62296877 -3.37455871 -2.89841583
-2.71511943 -3.15084733 -3.37455871 -3.47393365 -3.47393365 -3.37455871
-3.15084733 -2.71511943 -2.36299746 -2.71511943 -2.89841583 -2.98107006
-2.98107006 -2.89841583 -2.71511943 -2.36299746]

Phi of Right Face(face_5): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Phi of Left Face(face_6) : [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Added Mass........
Surge Added mass : 420.9864239929449

Acknowledgement

A course named **Numerical Ship and Offshore Hydrodynamics** taught in our department by eminent **Dr. Ranadev Datta** , is useful for this project.

Bibliography

1. Newman, J. N., 1977, Marine Hydrodynamics, MIT Press
2. https://en.wikipedia.org/wiki/Added_mass
3. https://sites.google.com/view/marine-hydrodynamics/home