Table of Contents

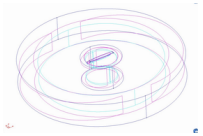
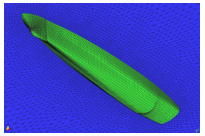
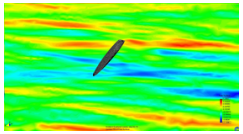
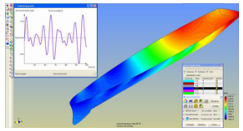
Seakeeping Reference

It is possible to obtain help for several items in the data tree and windows simply by passing the mouse pointer over them.

1.1 Introduction

FEM-Seakeeping is an advanced tool available for analyzing seakeeping problems developed at the International Center for Numerical Methods in Engineering (www.cimne.com) in colaboration with CompassIS (www.compassis.com). Despite of the youth of this software, it is quickly gaining recognition among research institutions, universities and companies. While most seakeeping softwares available are based on the frequency domain and boundary element method, FEM-Seakeeping has been conceived for more realistic simulations working in the time domain and using the finite element method, which allows the use of complex unstructured meshes recommended for the simulation of complex geometries.

FEM-Seakeeping is built in GID (www.gidhome.com), an universal, adaptive and user-friendly pre and postprocessor for numerical simulations in science and engineering. Moreover the software has been optimized such that just a few tens of thousands of nodes are necessary for simulating almost any existing floating offshore structure, reaching in many real time simulation and faster. This has been possible thanks to the development of a GPU library to enhance speeding up the simulations by carrying out the heavy calculation in graphic processing units.

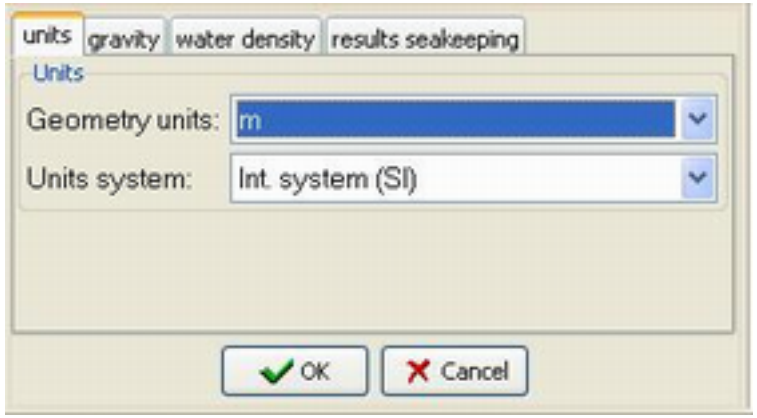





FEM-Seakeeping also offers the capability of introducing any external loads acting over the structure under study, other than those induced by waves. Moreover, prescribed pressure variation can be imposed over selected free surfaces, allowing simulating devices such as wave energy converters based on the oscillating water column principle, as well as aircushioned vessels. Furthermore, FEM-seakeeping results are prepared to be read by RamSeries, a FEM based structural program specialized in structural analysis and design of marine structures. This multiphysics platform allows the simulation of complex designs such as tension leg plattforms used for floating wind turbines.

1.2 General Data

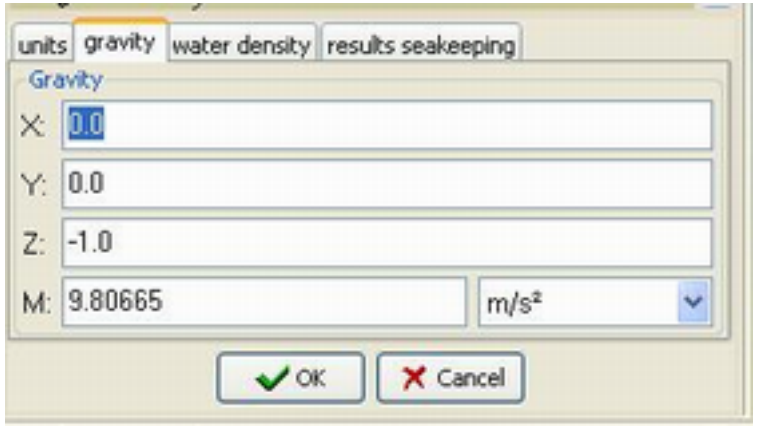
Set the Units, gravity value and Water density.

Units:



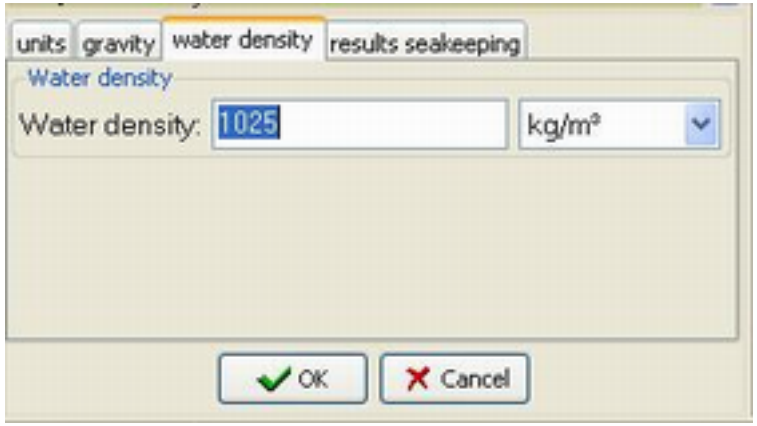
Gravity:

Set the gravity value and the direcction as well as the metric unit.



Water density:

Introduce the water density and the metric unit.



Results Seakeeping:

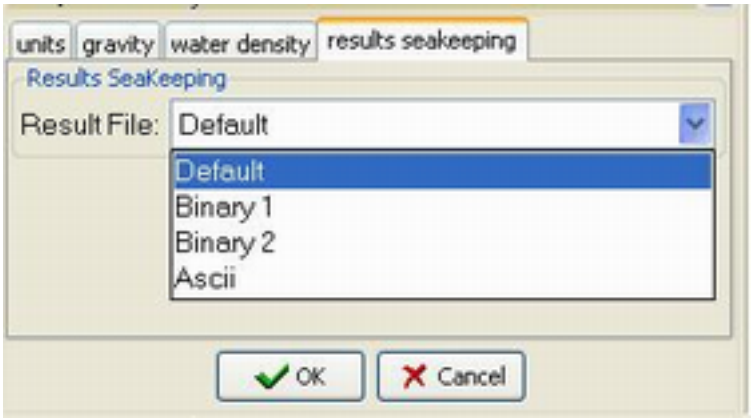
Select the format of the output file. The options available are:

- Default: will write the output file using ASCII format for older postprocessing.

- Binary 1: will write the output file using a binary format for older postprocessing.

- Binary 2: will write the output file using a binary format for newer postprocessing.

- ASCII: will write the output file using ASCII format for older postprocessing.



1.3 Problem description

Set the environment bathymetry, wave absorption, and wave radiation across the edge of the computational domain:

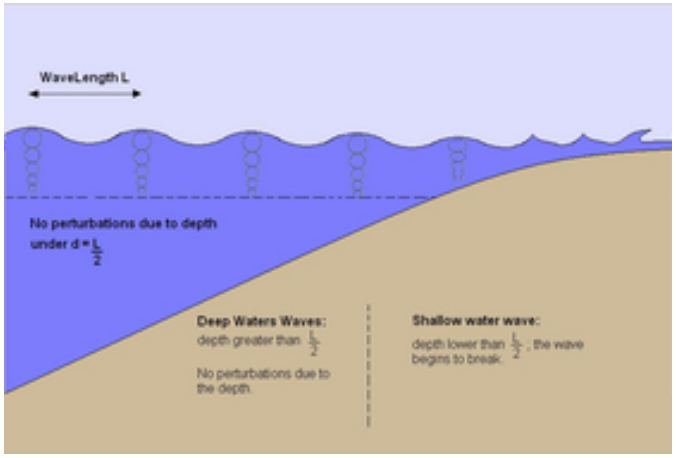

Tdyn Data
Simulation Type
General Data
Problem description
Bathymetry: Infinite depth
Depth: 0.0 m
Wave absorption: yes
Absorption factor: 1.0
Beach: 1.0 m
Sommerfeld radiation condition: Yes
Wave environment
Time analysis
Numerical setup
Body data
Initial Conditions
Boundary conditions
Problem description
Bathymetry:
Infinite depth
Depth:
0.0
m
Wave absorption
Absorption factor:
1.0
Beach:
1.0
m
Sommerfeld radiation condition
OK
Cancel

- **Bathymetry**

- **Infinite depth**: to be used when the depth is much larger than the wave lengths. In this case, the depth of the domain should be at least about half the wave length of the largest wave length.

- **Constant depth**: to be used when the bottom is flat, and the depth is constant and smaller than the wave lengths. If the water depth is larger than the wave lengths, it is recommended to use the infinite depth option.

- **Depth**: only available if "Bathymetry=Constant depth" was selected. Introduce the depth value to be used.

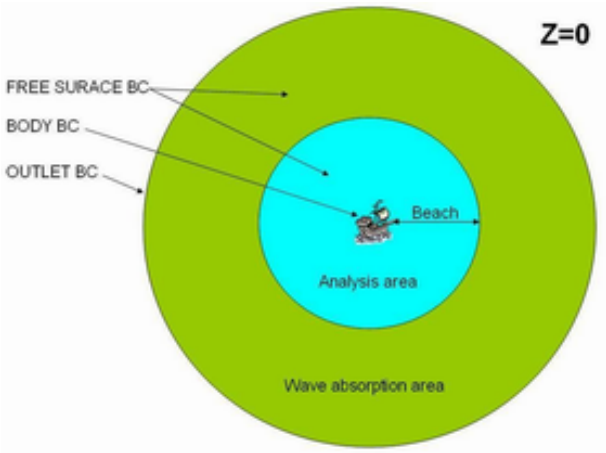


* Note that when the depth is greater than $\frac{L}{2}$ the wave has no perturbation due to the depth, but it changes considerably when the depth is less than $\frac{L}{2}$ and greater than $\frac{L}{20}$, in shallow waters the wave begins to break.

- **Wave absorption**: select if scattered waves generated by the presence of the body are to be absorbed when moving away from the body.

- **Absorption factor**: determines how strong the dissipation is (recommended value "1"). Large absorption factors might cause instabilities and wave reflection.

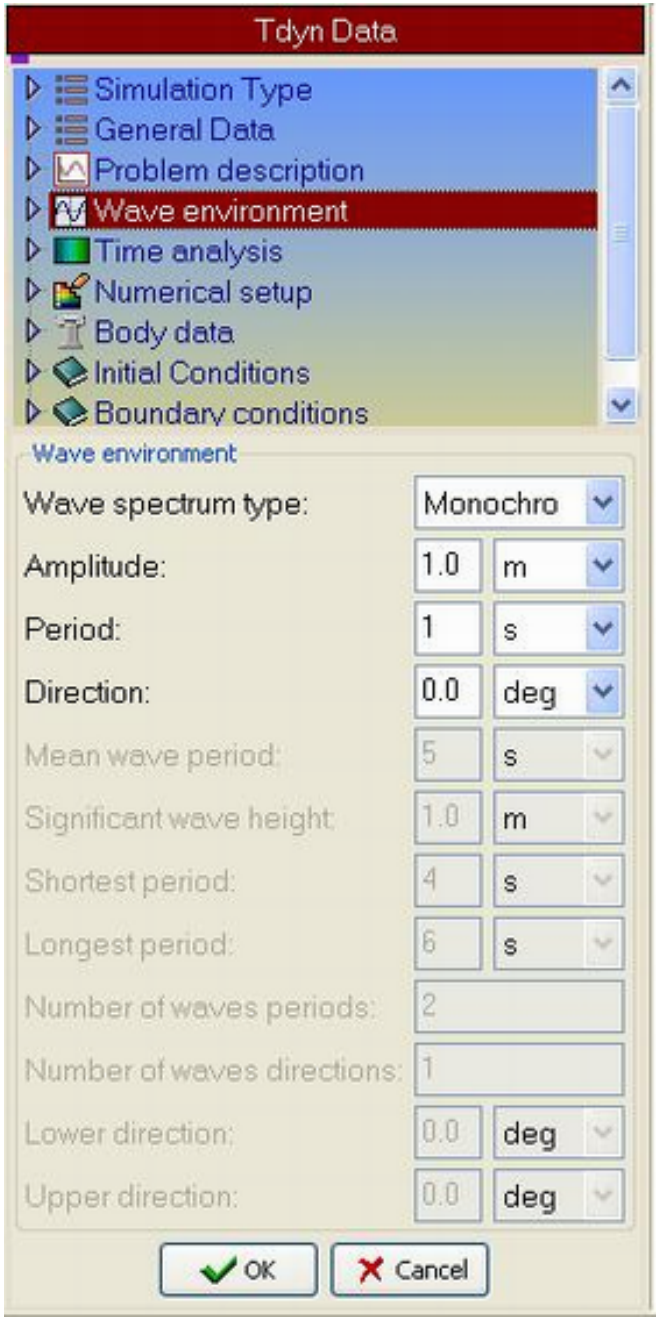
- **Beach**: Determine how far from the gravity center of the body the free surface absorption starts. For instance, if Beach=100, there will be no absorption within the circle of center at (XG,YG,0) and radius 100m. It is recommended the outer boundary of the domain to be circular and separated from the body at least 1 wavelength.



- **Sommerfeld radiation condition**: Select if scattered waves are to leave the computational domain throught the outlet boundaries.

1.4 Wave environment

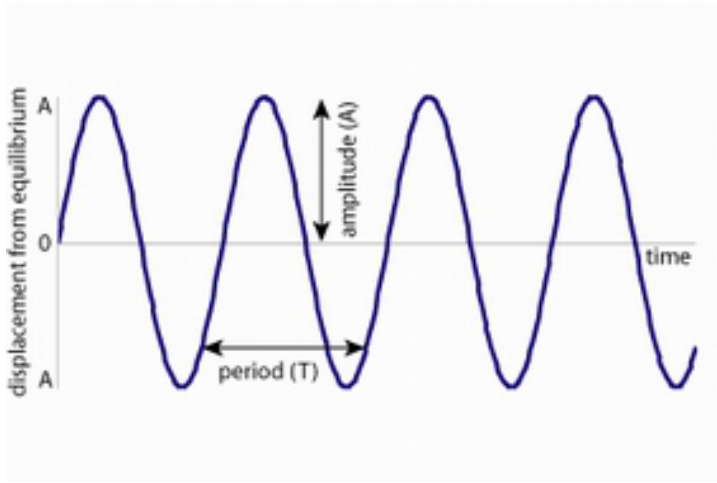
Set the type of wave environment.


Tdyn Data
Simulation Type
General Data
Problem description
Wave environment
Time analysis
Numerical setup
Body data
Initial Conditions
Boundary conditions
Wave environment
Wave spectrum type:
Monochro
Amplitude:
1.0
m
Period:
1
s
Direction:
0.0
deg
Mean wave period:
5
s
Significant wave height:
1.0
m
Shortest period:
4
s
Longest period:
6
s
Number of waves periods:
2
Number of waves directions:
1
Lower direction:
0.0
deg
Upper direction:
0.0
deg
OK
Cancel

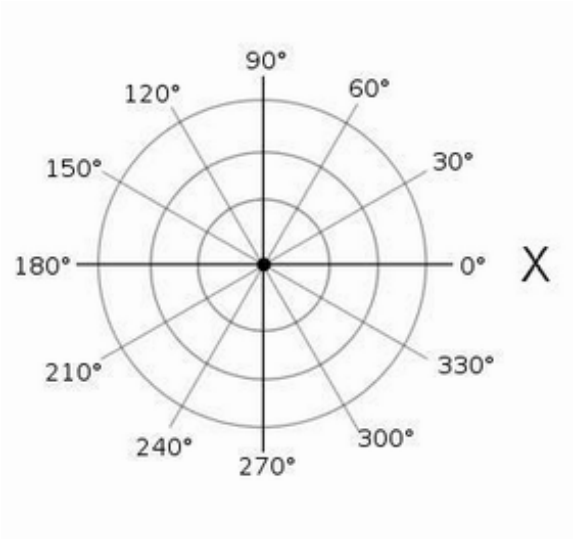
- **Monochromatic wave**

This option generates a wave environment that corresponds to a monochromatic wave. Inputs:

- **Wave amplitude and period:**



- **Direction:**



*Note that the direction is measured from the X-axis and in an counterclockwise direction.

- **Pearson Moskowitz**

This option will set the wave environment to that correspondint to one realization of the Pierson Moskowitz spectrum. The Pierson and Moskowitz assumed that if the wind blew steadily for a long time over a large area, the waves would come into equilibrium with the wind. This is the concept of a fully developed sea. Here, a long time is roughly ten-thousand wave periods, and a "large area" is roughly five-thousand wave-lengths on a side.

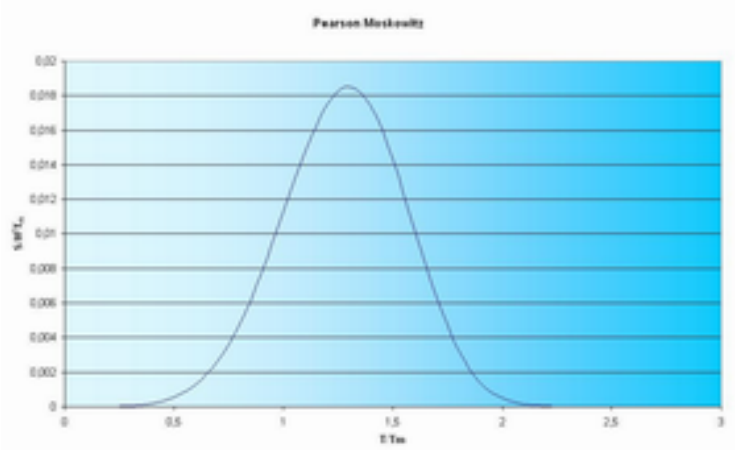
The Pierson Moskowitz spectrum is given by:

$$\frac{S(\varpi)}{H_s^2 T_m} = \frac{0.11}{2\pi}\left(\frac{\varpi_0}{\varpi}\right)^5 \exp\left[-0.44\left(\frac{\varpi_0}{\varpi}\right)^4\right]$$

where $S(\varpi)$ is the wave energy density; H_s is the significant wave height; T_m is the mean period in seconds; $\varpi = 2\pi/T$; T is the wave period in seconds; and

$\varpi_0 = 2\pi/Tm$. The Pierson-Moskowitz formula in terms of Period:

$$\frac{S(T)}{H_s^2 T_m} = \frac{0.11}{2\pi}\left(\frac{T}{T_m}\right)^5 \exp\left[-0.44\left(\frac{T}{T_m}\right)^4\right]$$



This option generates a wave environment that corresponds to an random realization of the Pearson Moskowitz spectrum. Wave phases are imposed randomly. Inputs:

- **Mean wave period (Tm).**
- **Significant wave height (H_S).**
- **Shortest wave period (Tmin):** Tmin=Tm/2,2 recommended.
- **Longest wave period (Tmax)**: Tmax=2,2xTm recommended.
- **Number of wave periods**: or number of wave frequencies to be used.
- **Number of wave directions**: in case the waves propagate within an angular sector, this parameter determines in how many directions the angular sector will be discretized.

- **Lower direction**: indicates the lower limit of the angular sector respect to the x axis.

- **Upper direction**: indicates the upper limit of the angular sector respect to the x axis.

* Note: the total number of waves used in the realization will be the "number of wave periods" times "Number of wave directions".

- **White noise**: this option generates a wave environment that corresponds to a realization of a white noise. Wave phases are imposed equal to zero. Inputs:

- **Amplitud**e: all waves will have the same amplitude

- **Direction**: all waves will propagate in the same direction. 0º means along x axis.

- **Shortest period**: minimum wave period to be considered.

- **Longest period**: maximum wave period to be considered.

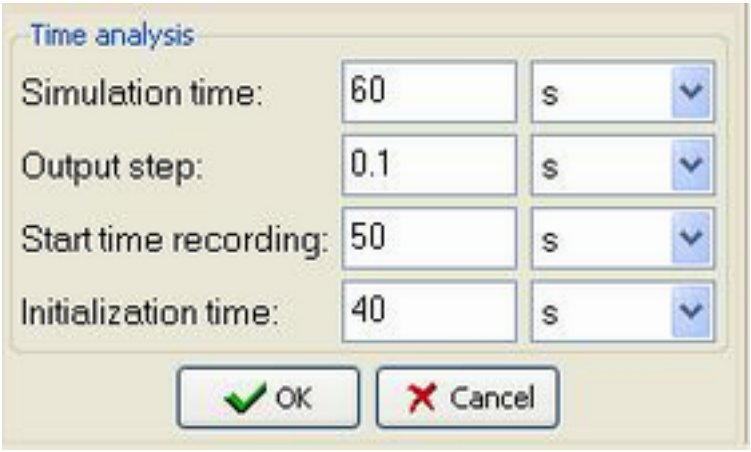
- **Number of wave periods**: or number of wave frequencies to be used.

* Note: this realization is done with a set of waves of equal amplitude and direction, wave phases set to zero, and periods uniformly distributed between the minimum and maximum periods.

1.5 Time analysis

- **Simulation time**: set the length of the simulation.
- **Output step**: set the lag-time between recordings. Should the output step be shorter than the time step, the former would become equal to the latter.
- **Start time recording**: set the point in time when recording of results will start.

- **Initialization time**: set the length in time to initialized the problem.



* Note: During the initialization time the interaction between the wave environment and the body will be introduced smoothly to avoid unrealistic initial situations.

1.6 Numerical setup

- **Processor Unit**: Options:

-**CPU**: all calculations are carried out in the CPU.

-**CPU+GPU**: solver computations will be carried out in the graphic processing unit if possible. All the other computations are carried out in the CPU.

- **Solver**: Select the solver to be used. The deflated conjugate gradient is recommended. In case it doesnt work, Conjugate gradient is recommended next.

Preconditioner: Select the preconditioner to be used.

- It is recommended to select ILU if the Processor Unit is set to CPU.

- It is recommended to select SPAI if the Processor Unit is set to CPU + GPU. If calculations seems to be too slow, switch to diagonal preconditioner and see which one goes runs faster.

- **Solver tolerance**: maximum tolerance allowed before convergence is reached.

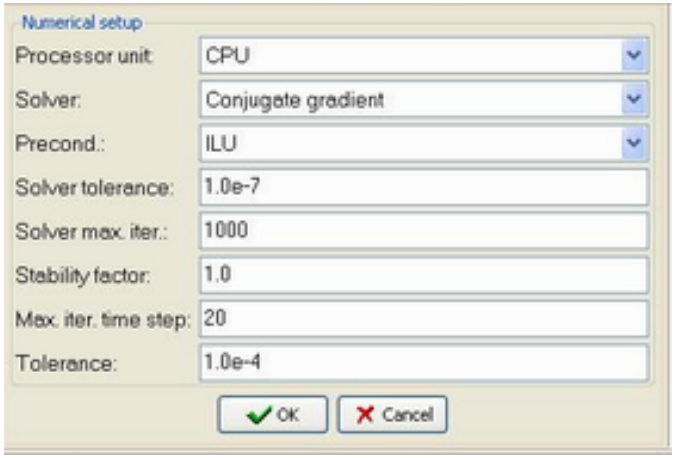
- **Solver max iterations**: maximum number of iterations to be carried out by the solver if convergence is not reached.

- **Stability factor**: this factor controls the time step to ensure stability.

*Note: the lower the stability factor is, the lower the time step will be, and the more stable the scheme will behave. It is recommended to start with a factor of 1.

- **Max iterations time step:** maximum number of iterations in one time step to reach convergence of the algorithm which calculates the object dynamics coupled with the wave resolution.

- **Tolerance:** maximum error allowed in the previous algorithm.



1.7 Body data

Body properties

- **Mass** : introduce the mass of the object. For freely floating objects, it is useful to introduce the mass as a function: "vol*density" or "disp"; which is equal to the displacement.
- **XG** : x coordinate of the gravity center of the floating body.
- **YG**: y coordinate of the gravity center of the floating body.
- **ZG**: z coordinate of the gravity center of the floating body.
- **Radii of gyration**: the elements of the Inertial matrix are related to the radii of gyration as: Iii=Mass*rii*rii; Pij=Mass*rij*|rij|. Then:

  rxx: rxx=sqrt(Ixx/Mass)

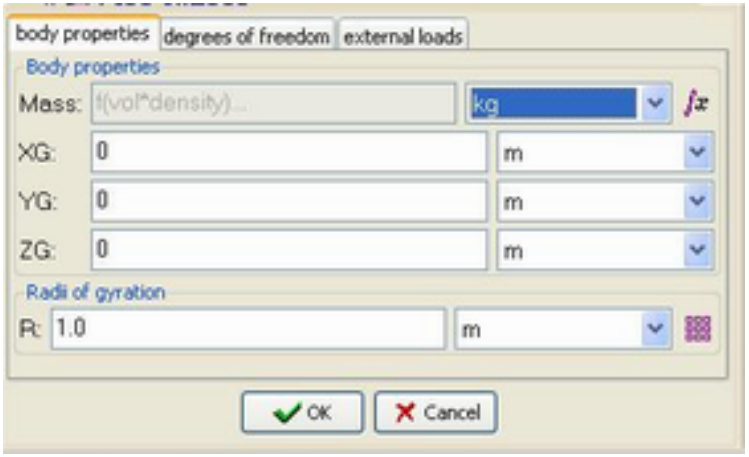
  rxy: rxy=(Pxy/|Pxy|)*sqrt(|Pxy|/Mass)

  rxz: rxz=(Pxz/|Pxz|)*sqrt(|Pxz|/Mass)

  ryy: ryy=sqrt(Iyy/Mass)

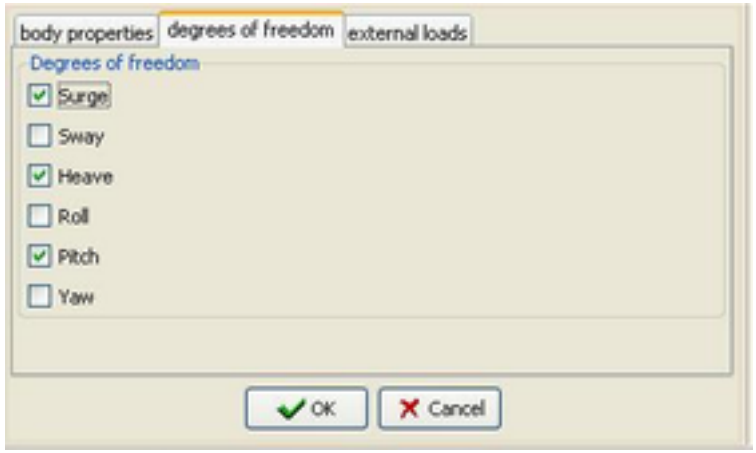
  ryz: ryz=(Pyz/|Pyz|)*sqrt(|Pyz|/Mass)

rzz: rzz=sqrt(Izz/Mass)

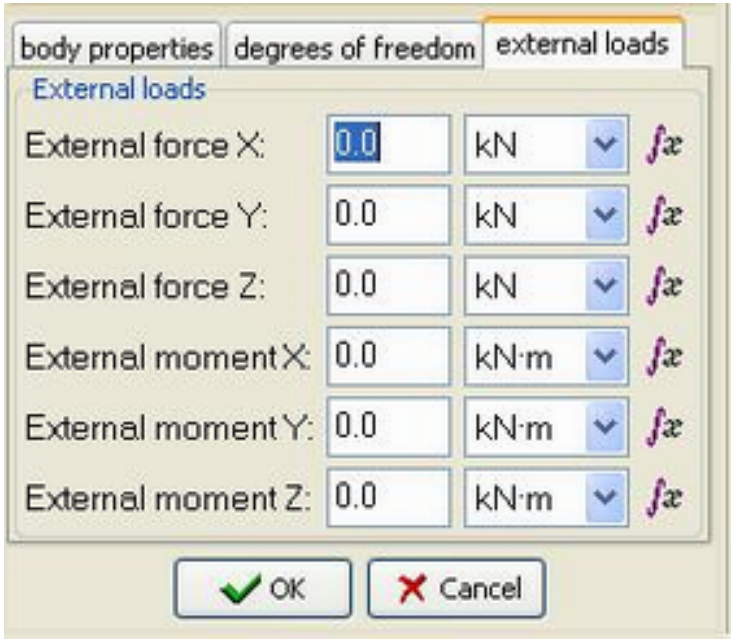


Degrees of freedom

- **Surge**: select if the floating object is supposed to translate along the x direction.
- **Sway**: select if the floating object is supposed to translate along the y direction.
- **Heave**: select if the floating object is supposed to translate along the z direction.
- **Roll**: select if the floating object is supposed to rotate around the x direction.
- **Pitch**: select if the floating object is supposed to rotate around the y direction.
- **Yaw**: select if the floating object is supposed to rotate around the z direction.



External Loads



External Loads can be defined by a set of functions representing forces and

moments other than the pressure forces induced by waves and hydrostatic restoring forces and moments. The following variables can be used to define the functions:

- mass: is the mass of the body under study.
- xg: x coordinate of the center of gravity of the body.
- yg: y coordinate of the center of gravity of the body.
- zg: z coordinate of the center of gravity of the body.
- Ijj: Diagonal elements of the inertial matrix respect to the gravity center of the body.
- Kij: Element of the hydrostatic restoring Matrix.
- dx, dy, dx: displacement (translation) of the center of gravity of the body (x,y,z components, surge, sway and heave). The displacement of an arbitrary point xi, yi, zi of the body can be used by dx[xi,yi,zi], dy[xi,yi,zi], dz[xi,yi,zi].
- rx, ry, rx: rotation of the body (x,y,z components, roll, pitch and yaw).
- vx, vy, vz: velocity of the translation of the body (x,y,z components, surge, sway and heave).
- wx, wy, wz: velocity of the rotation of the body (x,y,z components, roll, pitch and yaw).
- t: physical time of the analysis.
- *pi* : 3.1415926535897932385
- *exp(1)* : 2.7182818284590452354
- AreaP: surface area in which is applied the PfreeSurface condition.
- Vol: water displacement volume.
- density: substance density.
- disp: Object displacement. Water displaced weight.
- Pave: Surface average pressure in which is applied free surface condition.
- P: Total pressure applied to free surface condition.
- Pym: y coordinate of the center of mass of the free surface with prescribed

pressure.

- Pxm: x coordinate of the center of mass of the free surface with prescribed pressure.
- Flux: air flow displaced by the free surface assigned with PFreeSurface BC.

Example:

1.2*dx+0.1*dx^2+1.2*dy+0.1*dy^2

0.5*dx[0.5,1.0,0.0]+0.5*dy[0.5,1.0,0.0]+0.2*dz

Furthermore, the following function operators are available:

- *sqrt* : the sqrt function calculates the square root of the argument. Syntax: sqrt(.)
- *abs* : the abs function calculates the absolute value of the argument. Syntax: abs(.)
- *ln* : logarithm of the argument, e base. Syntax: ln(.)
- *log* : logarithm of the argument, decimal base. Syntax: log(.)
- *fac* : factorial of the argument. Syntax: fac(.)
- *sin* : sine of the argument. Syntax: sin(.) (argument given in radians).
- *cos* : cosine of the argument. Syntax: cos(.) (argument given in radians).
- *tan* : tangent of the argument. Syntax: tan(.) (argument given in radians).
- *asin* : The asin function returns the arcsine of the argument in the range -π/2 to π/2 radians. Syntax: asin(.).
- *acos* : The acos function returns the arccosine of the argument in the range 0 to À radians. Syntax: acos(.).
- *atan* : The atan function returns the arctangent of the argument in the range -π/2 to π/2 radians. Syntax: atan(.) (result given in radians).
- *sinh* : hyperbolic sine of the argument. Syntax: sinh(.).
- *cosh* : hyperbolic cosine of the argument. Syntax: cosh(.).
- *tanh* : hyperbolic tangent of the argument. Syntax: tanh(.).

- *exp* : the exp function calculates the exponential value of the argument. Syntax: exp(.).
- *Interpolate* : performs a linear interpolation, based on the given data. Two arguments are required: a list of pairs (ξ,η), defining a polylineal curve, and a function defining the point (ξ) where the evaluation is to be done. Syntax: interpolate(#ξ_1,η_1,ξ_2,η_2,ξ_3,η_3,...#.).
- *InterpolateSpline* : performs a spline interpolation, based on the given data. Two arguments are required: a list of pairs (ξ,η), defining a the curve, and a function defining the the point (ξ) where the evaluation is to be done. Syntax: interpolatespline(#ξ_1,η_1,ξ_2,η_2,ξ_3,η_3,...#.).
- *srand* : The rand function returns a pseudorandom integer in the range 0 to 1, based on the argument given as seed. Syntax: srand(.).
- *int* : Integer conversos. Syntax: int(.).
- - : change sign operator. Syntax: (-expression).
- *j0* : Calculates Bessel function of first kind and order 0, at the given point. Syntax: j0(.).
- *j1* : Calculates Bessel function of first kind and order 1, at the given point. Syntax: j1(.).
- *jn* : Calculates Bessel function of first kind and order n, at the given point. Syntax: jn(.,.), where the first argument is the evaluation point and the second is the order of the Bessel function.
- *y0* : Calculates Bessel function of second kind and order 0, at the given point. Syntax: y0(.).
- *y1* : Calculates Bessel function of second kind and order 1, at the given point. Syntax: y1(.).
- *yn* : Calculates Bessel function of second kind and order n, at the given point. Syntax: yn(.,.), where the first argument is the evaluation point and the second is the order of the Bessel function.

Examples:

1.2*sin(dx)+dx^2

abs(dx)+0.5*abs(dy)-0.01*vx

The operators that can be used for functions definitions are:

- + : adding operator.

Syntax: [adding_expression] + [adding_expression].

- - : substraction operator.

Syntax: [substraction_expression] - [substraction_expression].

- ^ : exponent operator.

Syntax: [exponent_expression] ^ [function_expression].

- * : multiplicative operator.

Syntax: [multiplicative_expression] * [multiplicative _expression].

- / : division operator.

Syntax: [multiplicative_expression] / [quotient_expression].

- *div* : integer division operator int(x/y+0.5).

Syntax: ([multiplicative_expression]) div ([quotient_expres-sion]). Example: (x)div(2+y).

- *idiv* : integer division operator int(x/y+0.5). Similar to div operator but with different syntax.

Syntax: idiv ([multiplicative_expression], [quotient_expres-sion]). Example: idiv(x,2+y).

- *mod* : integer division module operator int(x+0.5)%int(y+0.5).

Syntax: ([multiplicative_expression]) mod ([quotient_expres-sion]). Example: (t)mod(2).

- *imod* : integer division module operator int(x+0.5)%int(y+0.5). Similar to mod operator but with different syntax.

Syntax: imod ([multiplicative_expression],[quotient_expres-sion]). Example: imod(t,2).

- *rdiv* : real division operator int(x/y).

Syntax: ([multiplicative_expression]) rdiv ([quotient_expres-sion]). Example: (t)rdiv(5).

- *ddiv* : real division operator int(x/y). Similar to rdiv operator but with different syntax.

Syntax: ddiv ([multiplicative_expression], [quotient_expres-sion]). Example: ddiv(t,5).

- *rmod* : real division module operator x/y-int(x/y).

Syntax: ([multiplicative_expression]) rmod ([quotient_expres-sion]). Example: (t)rmod(5).

- *dmod* : real division module operator x/y-int(x/y). Similar to rmod operator but with different syntax.

Syntax: dmod ([multiplicative_expression], [quotient_expres-sion]). Example: dmod(t,5).

- *max* : maximum operator.

Syntax: max ([expression], [expression]). Example: max(x,y).

- *min* : minimum operator.

Syntax: min ([expression], [expression]). Example: min(x,y).

- *not* : not operator.

Syntax: not([function_expression]).

- ~ : not operator.

Syntax: ~([function_expression]).

Examples:

(2*y)

(5*(y+1))/2

(z*y)mod(5)

imod(z*y) (5)

(5^4)

The relational (binary) operators compare their first operand with their second operand to test validity of the specified relationship. The result of the relational expression is 1 if the tested relationship is true and 0 if it is false. The binary operators that can be used for functions definitions are:

- < : less than operator.

Syntax: [expression] < [expression].

- < = : less or equal than operator.

Syntax: [expression] <= [expression].

- >= : greater or equal than operator.

Syntax: [expression] >= [expression].

- > : greater than operator.

Syntax: [expression] > [expression].

- = : equal operator.

Syntax: [expression] = [expression].

- ~= : not equal operator.

Syntax: [expression] != [expression].

- & : and operator.

Syntax: [expression] & [expression].

- | : and operator.

Syntax: [expression] | [expression].

Examples:

(y>2)

(x<=1)

(x!=1)

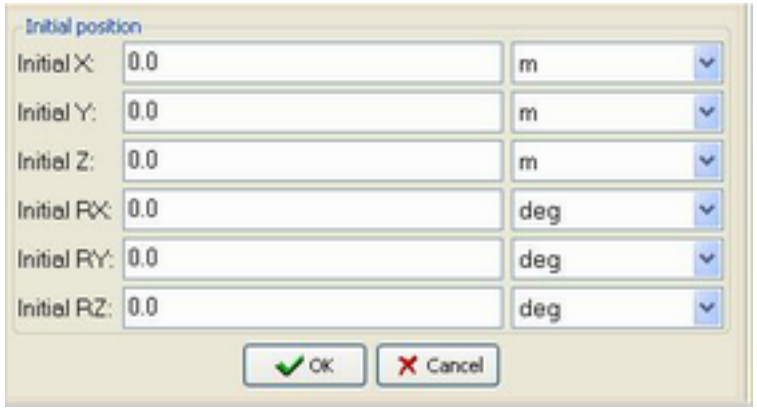
(y>2)&(x>2)&(x<3)&(y<3)

1.8 Initial conditions

Set the initial object conditions, the options available are:

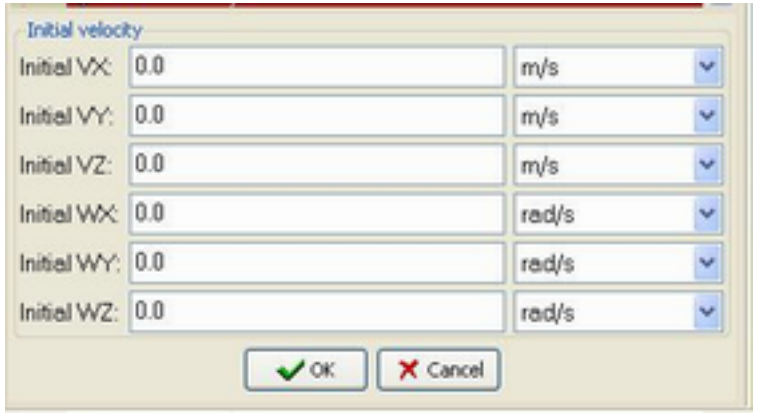
Initial Position

- Initial X: set a initial X position for the object, needed to check Surge in 'Degrees of freedom' .
- Initial Y: set a initial Y position for the object, needed to check SWay in 'Degrees of freedom'.
- Initial Z: set a initial Z position for the object, needed to check Heave in 'Degrees of freedom'.
- Initial RX: set a initial RX rotation angle for the object, needed to check Roll in 'Degrees of freedom'.
- Initial RY: set a initial RY rotation angle for the object, needed to check Pitch in 'Degrees of freedom'.
- Initial RZ: set a initial RZ rotation angle for the object, needed to check Yaw in 'Degrees of freedom'.



Initial Velocity

- Initial VX: set a initial X velocity for the object, needed to check Surge in 'Degrees of freedom'.
- Initial VY: set a initial Y velocity for the object, needed to check SWay in 'Degrees of freedom'.
- Initial VZ: set a initial Z velocity for the object, needed to check Heave in 'Degrees of freedom'.
- Initial WX: set a initial WX angular velocity for the object, needed to check Roll in 'Degrees of freedom'.
- Initial WY: set a initial WY angular velocity for the object, needed to check Pitch in 'Degrees of freedom'.
- Initial WZ: set a initial WZ angular velocity for the object, needed to check Yaw in 'Degrees of freedom'.

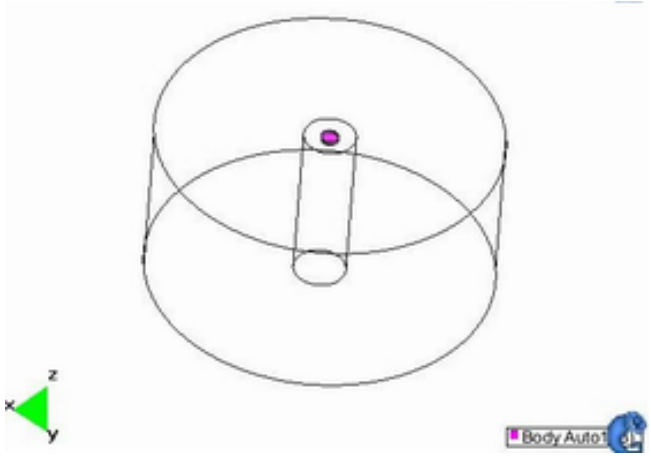


1.9 Boundary conditions

Conditions applied to different surfaces depending in their meaning.

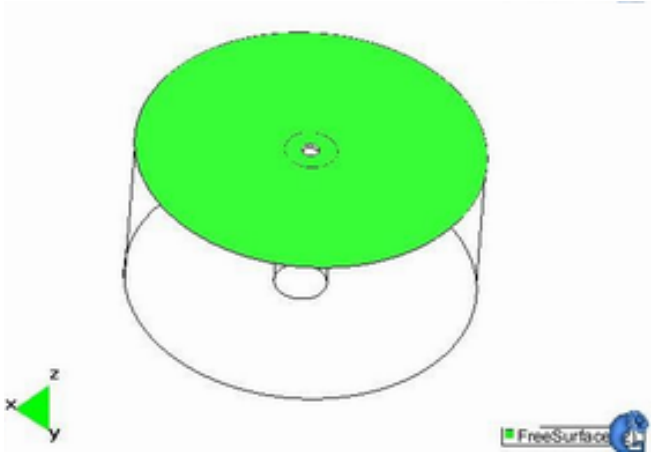
- **Body**

Condition to be applied on the surfaces of the body under study. Next figure shows the selected body BC for the case study of a freely floating cylinder:



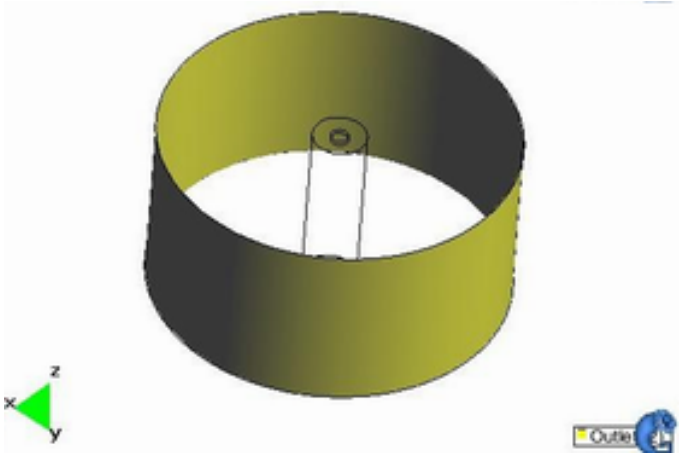
- **Free Surface**

Condition to be applied to the free surface of the computational domain. See figure:



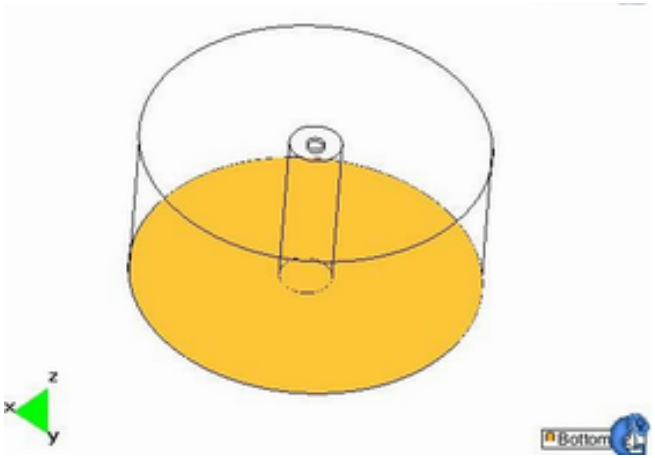
- **Outlet**

Condition to be applied to the exterior surfaces of the computational domain.



- **Bottom**

Condition to be applied to the floor of the computational domain.



- **Wall**

Boundary condition imposes no flow condition.

- **PfreeSuface**

Imposes pressure on the selected free surface. Should this condition be selected, the average pressure and pressure variation over the slected surfaces should be given. Any surface to be assigned this BC must be previously assigned the Free Surface BC as well.

1.10 Application example

Interaction between a floating body and a monochromatic wave:

Floating body: Cylindrical shape;

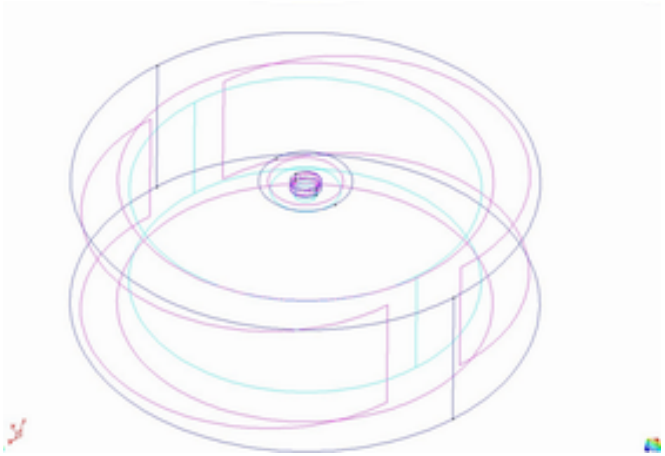
Radius	1 m
Draft	0.5 m

Incident wave data:

Amplitude	0.1 m
Period	8.971402 s
Direction	0 º
Depth	infinite

COMPUTATIONAL DOMAIN:

The computational domain is located in z<=0 and consist of the volume that extends within a cylinder of radius 15m and 10m in depth, and a inner cylinder representing the floating body.



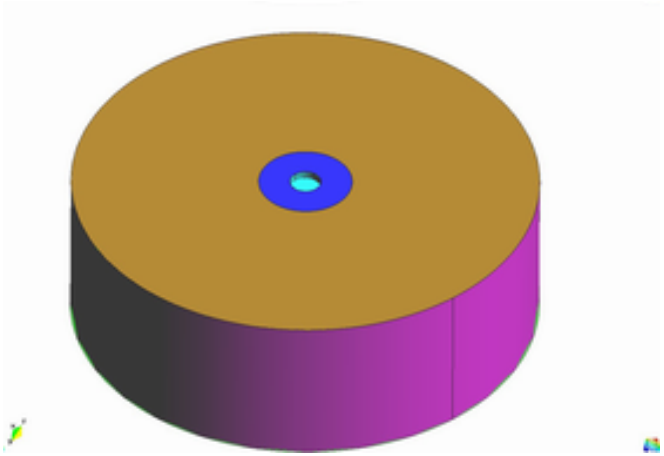
BOUNDARY CONDITIONS:

There are three types of boundary conditions to be applied:

Body: to be applied over the surfaces of the floating body (cyan)

Free surface: to be applied in those surfaces located at z=0 (blue and brown surfaces).

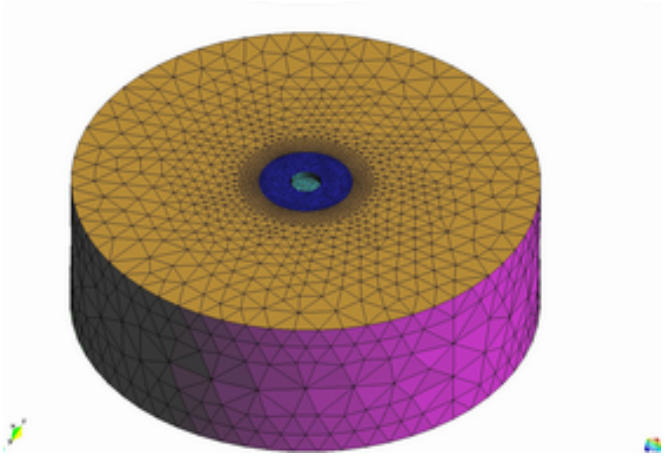
Outlet: To be applied over the outer surface. If a Sommerfeld radiation condition is to be applied, the outer surface should be of circular shape with center the origin of coordinates.



MESH GENERATION:

The mesh size has to be small enough to be able to accurately represent the floating body geometry as well as smaller than the minimum wave length to be considered. We set a element size of 0.2m at the body surfaces and at the blue free surface. The mesh size is allowed to grow in the outward direction. In order to avoid an unnecessary increase in the number of nodes, it is suggested to use element sizes in the order of:

- Outlet surface: 1/10th of the distance from the origin of coordinates.
- Bottom: 1/10th of the depth.



INPUTS:

GENERAL;

Unit_system=International_system,

gravity=9.80665,

density=1025,

;

PROBLEM_DESCRIPTION;

Bathymetry=Infinite_depth,

Wave_absorption=No,

Sommerfeld_radiation_condition=Yes,

;

WAVE_ENVIRONMENT;

Ambient_type=Monochromatic_wave,

Amplitude=1,

Period=8.9714,

Direction=0,

;

TIME_ANALYSIS;

Simulation_time=150,

Sampling_time=0.25,

Start_time_recording=100,

Initialization_time=80,

;

NUMERICAL_SETUP;

Solver=CONJUGATE_GRADIENT,

Preconditioner=ILU,

Solver_max_iterations=1000,

Solver_tolerance=1.0e-5,

beta=1.0,

Tolerance=1.0e-4,

Max_iter_time_step=20,

Processor_Unit=CPU,

;

EXTERNAL_LOADS;

MooringDX = "0;",

MooringDY = "0.0;",

MooringDZ = "0;",

MooringRX = "0.0;",

MooringRY = "0;",

MooringRZ = "0.0;",

;

SEAKEEPING;

Surge=Yes,

Sway=No,

Heave=Yes,

Roll=No,

Pitch=Yes,

Yaw=No,

Mass = "vol*density;",

XG=0,

YG=0,

ZG=0,

rxx=1

ryy=1

rzz=1

rxy=0.0

rxz=0.0

ryz=0.0

;

INITIAL_CONDITIONS;

Xinit=0,

Yinit=0,

Zinit=0,

RXinit=0,

RYinit=0,

RZinit=0,

VXinit=0,

VYinit=0,

VZinit=0,

WXinit=0,

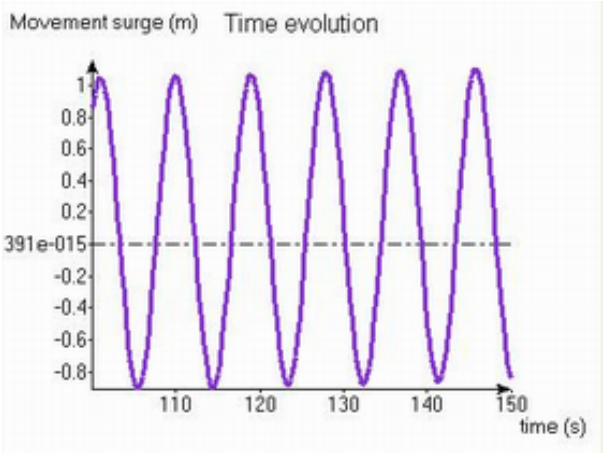
WYinit=0,

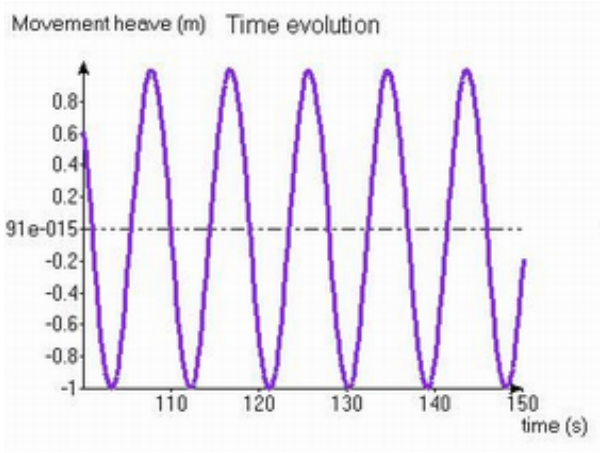
WZinit=0,

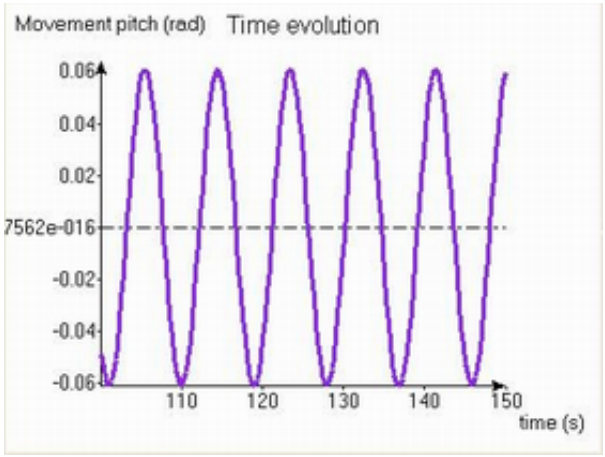
;

RESULTS:

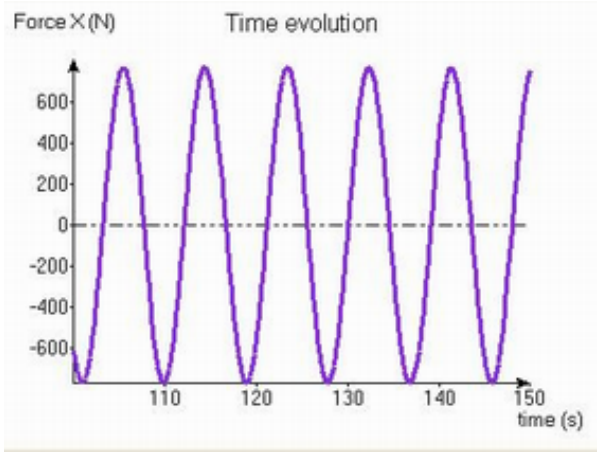
Body movements:

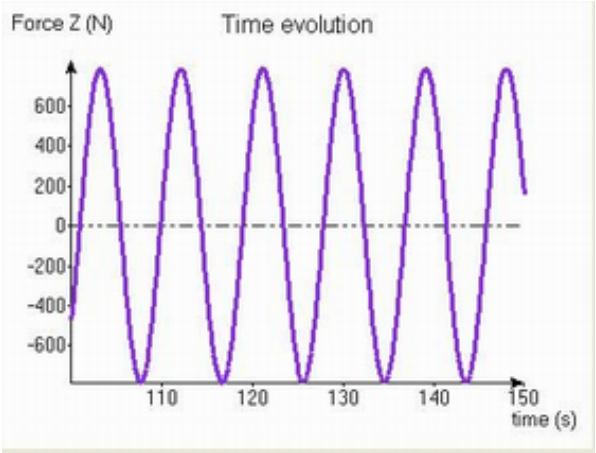

Movement surge (m)
Time evolution
1
0.8
0.6
0.4
0.2
391e-015
-0.2
-0.4
-0.6
-0.8
110
120
130
140
150
time (s)

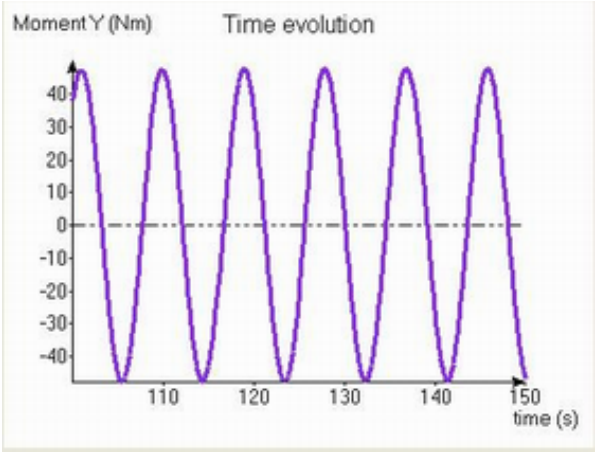

Movement heave (m)
Time evolution
0.8
0.6
0.4
0.2
91e-015
-0.2
-0.4
-0.6
-0.8
-1
110
120
130
140
150
time (s)

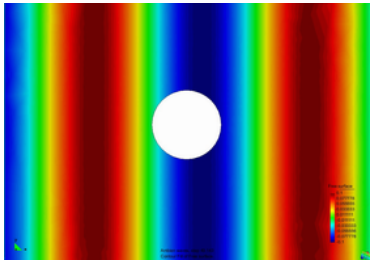


Body loads: obtained by integration of the pressure over the body surfaces.

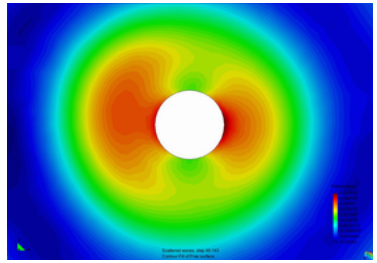



Force Z (N)
Time evolution
600
400
200
0
-200
-400
-600
110
120
130
140
150
time (s)

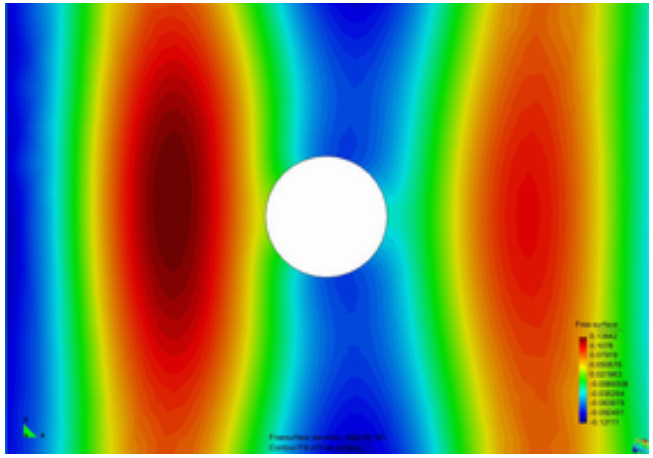

Moment Y (Nm)
Time evolution
40
30
20
10
0
-10
-20
-30
-40
110
120
130
140
150
time (s)



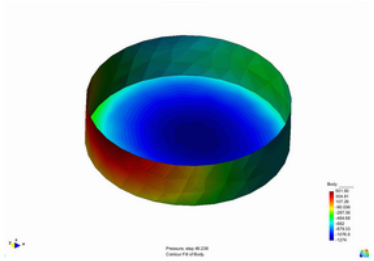
Incident wave



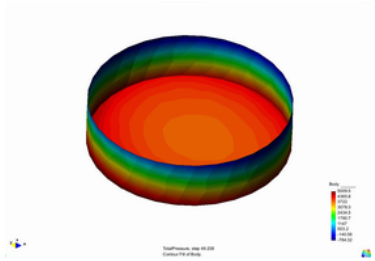
Scattered waves



Incident wave + scattered waves



Pressure induced



Pressure induced + hydrostatic pressure

1.11 Mathematical Model

Governing equations

We consider the first order diffraction-radiation problem of a floating body.

$$\nabla^2\varphi = 0 \text{ in } \Omega \qquad (1)$$

$$\delta_t\varphi + g\eta = -\frac{P}{\rho} + C \text{ in } z = 0$$

(dynamic free surface boundary condition) (2)

$$\delta_t\eta - \delta_z\varphi = 0 \text{ in } z = 0$$

(kinematic free surface boundary condition) (3)

$$\delta_z\varphi = 0 \text{ in } z = -H \qquad (4)$$

$$\nabla\varphi\, n_B = v_B \,.\, n_B \text{ in } \Gamma_B \qquad (5)$$

where φ and η are the first order potential and free surface elevation respectively; Ω is the fluid domain bounded by z = 0; P is the free surface pressure; ρ is the water density; C is a constant value; Γ_B represents the

wetted surface of a floating body; and H is the water depth. The domain is assumed to be infinite in the horizontal directions.

Velocity potential decomposition

The aim of this software is to simulate the dynamics of a floating body subjected to the action of waves. To do so, we will first model the environment as the sum of a number of airy waves. This can be expressed in terms of a velocity potential given by:

$$\Psi = \sum_m \frac{A_m g}{\varpi_m} \frac{\cosh(|k_m|(H + Z))}{\cosh(|k_m|H)} \cos(|k_m| x \cos ?_m + y \sin ?_m - \varpi_m t + \delta_m) \quad (6)$$

where A_m are the wave amplitudes; ϖ_m are the wave frequencies; k_m are the wave numbers; $?_m$ are the wave directions; and δ_m are wave phases. From this point on, we will refer to Ψ as the incident potential. This potential, along with the dispersion relation $\varpi_m^2 = g|k_m|\tanh(|k_m|H)$, fulfils Eqs. (1)-(4), and therefore is solution of the mathematical model in the absence of bodies.

In order to obtain the solution to the governing equations, we will use a velocity potential decomposition. Let φ be the solution to the governing equations. Then φ can be decomposed as $\varphi = \Psi + \Phi$, where Φ represents the velocity potential of waves diffracted and radiated by the body.

Introducing the velocity potential decomposition into the governing equations we obtained the equation to be fulfilled by Φ:

$$\nabla^2 \Phi = 0 \quad \text{in } \Omega \quad (7)$$

$$\delta_t\Phi + g\eta = -\frac{P}{\rho} + C' \text{ in } z = 0 \quad \text{(dynamic free surface boundary condition)} \quad (8)$$

$$\delta_t\eta - \delta_z\Phi = 0 \quad \text{in } z = 0 \quad \text{(kinematic free surface boundary condition)} \quad (9)$$

$$\delta_z\Phi = 0 \text{ in } z = -H \quad (10)$$

$$\nabla\Phi\, n_B = (v_B - \nabla\Psi) n_B \text{ in } \Gamma_B \quad (11)$$

Our purpose is to find Φ for a given incident potential and given v_B. To do so, we will solve Eqs. (7)-(11) in a finite domain by means of the finite element method.

Radiation condition and wave dissipation

Waves represented by Φ are born at the body and propagate in all directions away from the body. These waves have to either be dissipated or to be let go out the domain so they will not come back and interact with the body. Then, we will make use of a Sommerfeld radiation condition at the edge of the computational domain:

$$\delta_t\Phi + c\nabla\Phi n_B = 0 \quad \text{in } \Gamma_B \quad (12)$$

where Γ_B is the surface limiting of the domain in the horizontal directions, and c is a prescribed wave velocity. Eq. (12) will let waves moving at velocity c to escape out the domain. However, waves with very different velocities will not be leaving the domain. Then wave dissipation is introduced into the dynamic free surface boundary condition by varying the pressure such that:

$$\frac{P}{\rho} = P_0 + \kappa(x)\delta_z\Phi \tag{13}$$

where $\kappa(x)$ is a damping coefficient. Eq. (13) Increases pressure when the free surface is moving upward, while decrease the pressure when the free surface is moving downwards. By doing this, energy is transfer from the waves to the atmosphere and waves are damped. However, the coefficient $\kappa(x)$ will be set to zero in the area nearby the body so damping will no affect to the wave structure interaction.

Combining the dynamic and kinematic boundary condition, introducing Eq.(13), and adding Eq.(12), and choosing $C' = P_0$, the governing equations for Φ becomes:

$$\nabla^2\Phi = 0 \quad \text{in } \Omega \tag{14}$$

$$\delta_{tt}\Phi = -g\delta_z\Phi - \kappa(x)\delta_t\delta_z\Phi \quad \text{in } z = 0 \tag{15}$$

$$\delta_z\Phi = 0 \quad \text{in } z = -H \tag{16}$$

$$\nabla\Phi\, n_B = (v_B - \nabla\Psi)\, n_B \text{in } \Gamma_B \tag{17}$$

$$\delta_t\Phi + c\nabla\Phi\, n_R = 0 \text{ in } \Gamma_R \tag{18}$$

$$\eta = -\frac{1}{g}\delta_t\Phi - \frac{P_a}{\rho g} + \frac{C'}{g} \quad \text{in } z = 0 \text{ (kinematic free surface boundary condition)} \tag{19}$$

where the free surface elevation has been decoupled from the problem of obtaining the velocity potential.