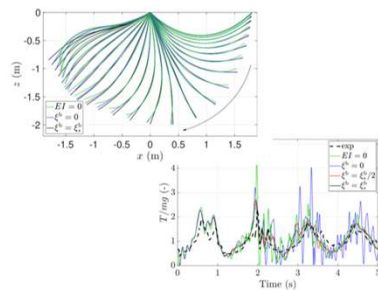


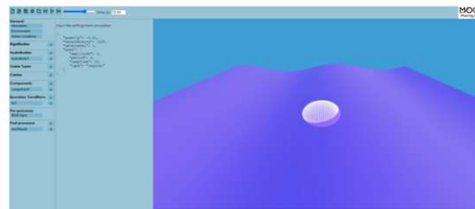
# MOODY SOFTWARE

## THE PARTS OF MOODY:



### MoodyCore

MoodyCore is the fundamental mooring solver/library based on a high-order discontinuous Galerkin method. In addition to cable dynamics MoodyCore also supports submerged rigid bodies and hydrodynamic bodies.



### MoodyMarine

MoodyMarine is the graphical user interface for the MoodyCore mooring and floating body library built with Electron.

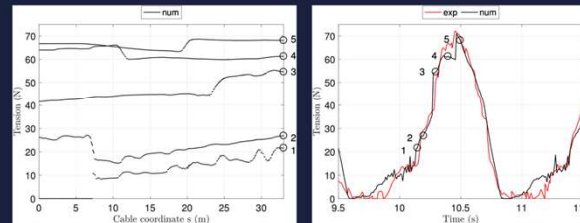


### MoodyAPI

MoodyAPI concerns the coupling of MoodyCore to hydrodynamic solvers like OpenFOAM's interFOAM solver.

# MOODY SOFTWARE – MOODYCORE

## Command line tools and Libraries



## MoodyCore

MoodyCore is the fundamental mooring solver/library based on a high-order discontinuous Galerkin method. In addition to cable dynamics MoodyCore also supports submerged rigid bodies and hydrodynamic bodies, as well as components such as tabulated spring-dampers.

Please note that at present MoodyCore (and hence MoodyMarine) does not support body-to-body interactions and constraints (like joints)

High-order DG

Bending Stiffness

Rigid Bodies

## Cable dynamics

- Nonlinear wave equation
- Hydrodynamic forces - Morison
- Bending stiffness
- Buoys and clump weights
- Springs and lookup-table components
- High-order DG-FEM method (C++)
- Explicit time-stepping

## Hydrodynamic bodies

- Cummins equation
- Direct integration of IRF
- Nemoh input style

## Pre- and post processing

- Converter between hydro formats
- Python/matlab plotting

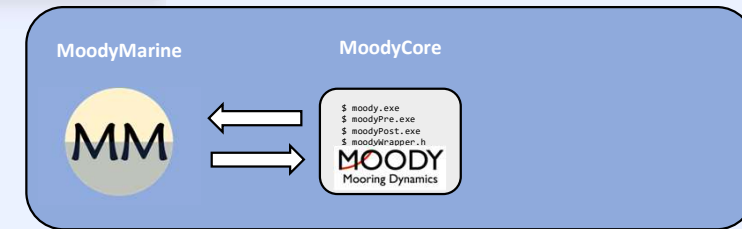
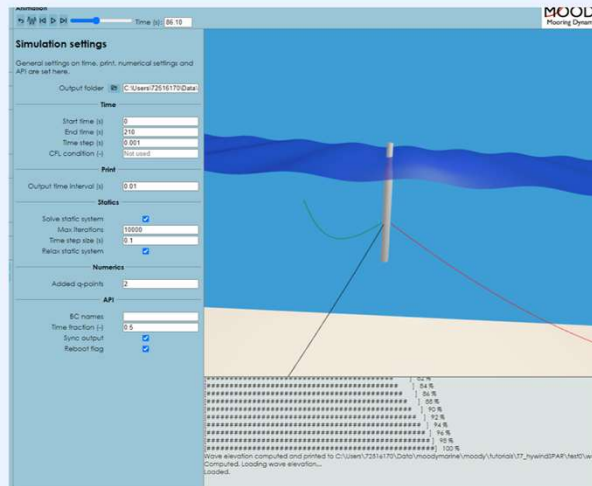
# MOODY SOFTWARE – MOODYMARINE

## MoodyMarine

MoodyMarine is the graphical user interface for the MoodyCore mooring and floating body library built with Electron.

It contains access to [NEHOM-v3](#) for computing hydrodynamic coefficients and has static and dynamic solver modes, as well as abilities to plot results and animations.

Windows Mac Linux



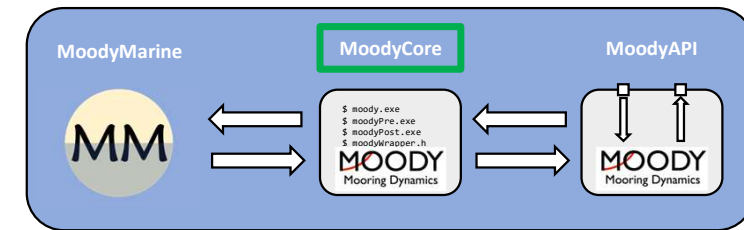
## MoodyCore Interface

- Input file generation
- Static evaluation
- Dynamic evaluation
- Wave visualization

## Nemoh interface

- Input file generation
- Simulation: Preproc, Solver, PostProc

# MOODY SOFTWARE – MOODYAPI



## Interfaces

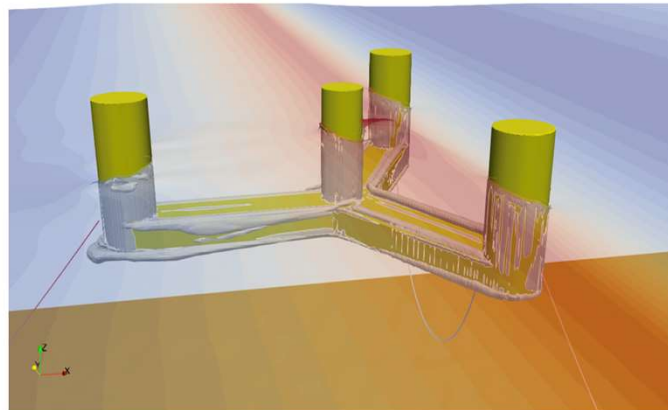
C++  
Python (in development)  
Fortran90

## Supports

OpenFOAM-v2212

## Existing but not “official”

OpenFOAM-v2106  
WEC-SIM  
FASTv7  
DualSphysics



## MoodyAPI

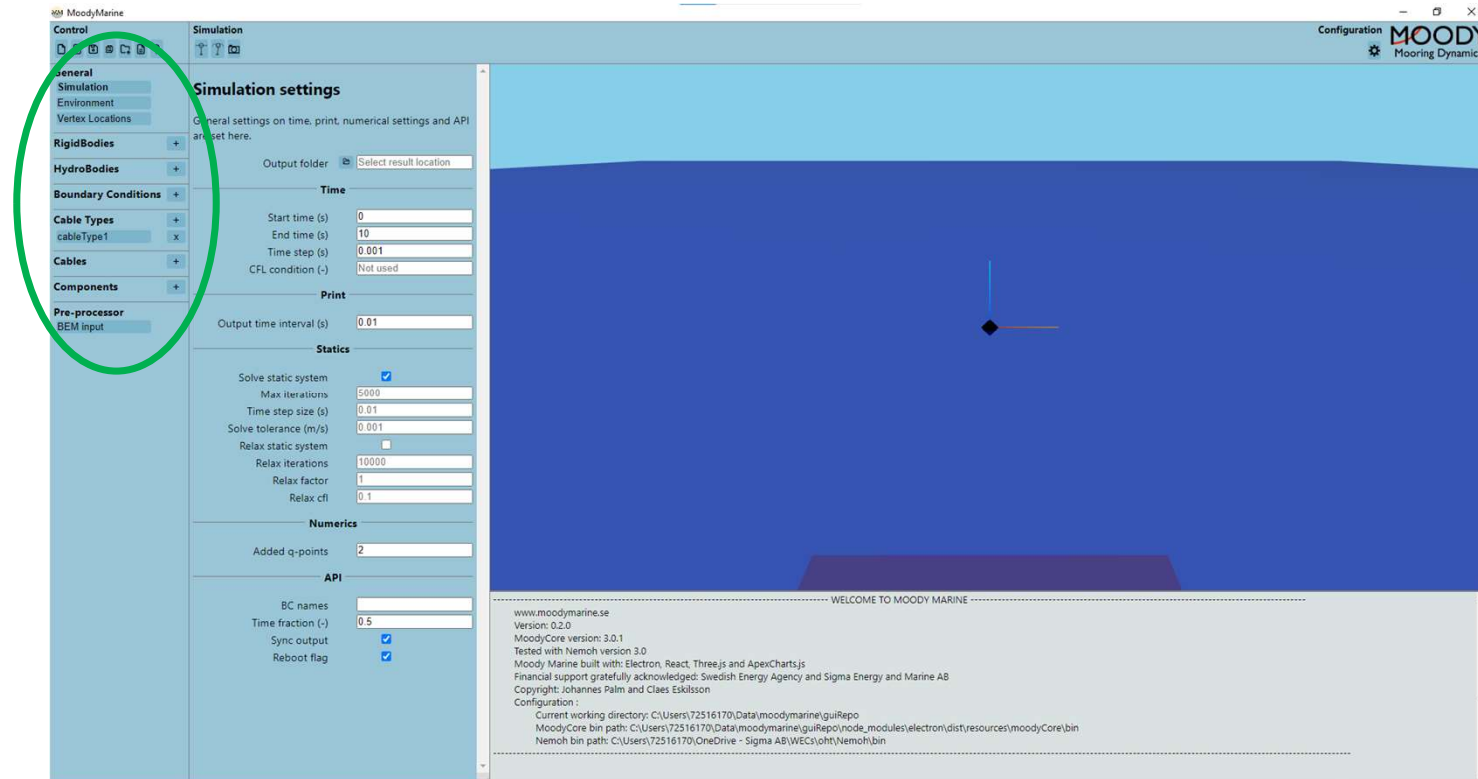
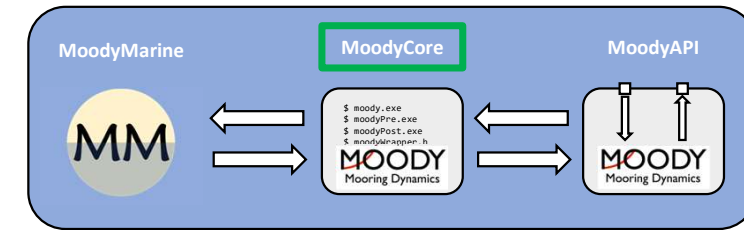
MoodyAPI concerns the coupling of MoodyCore to hydrodynamic solvers like OpenFOAM's interFOAM solver, giving the sixDoFRigidBody library access to dynamic mooring. In MoodyAPI can also be found additional tweaks of the sixDoFRigidBody library to include multi-bodies, relaxation of the mesh-motion for large translational motions, etc.

**Disclaimer:** This offering is not approved or endorsed by OpenCFD Limited, producer and distributor of the OpenFOAM software via [www.openfoam.com](http://www.openfoam.com), and owner of the OPENFOAM® and OpenCFD® trade marks.

OpenFOAM

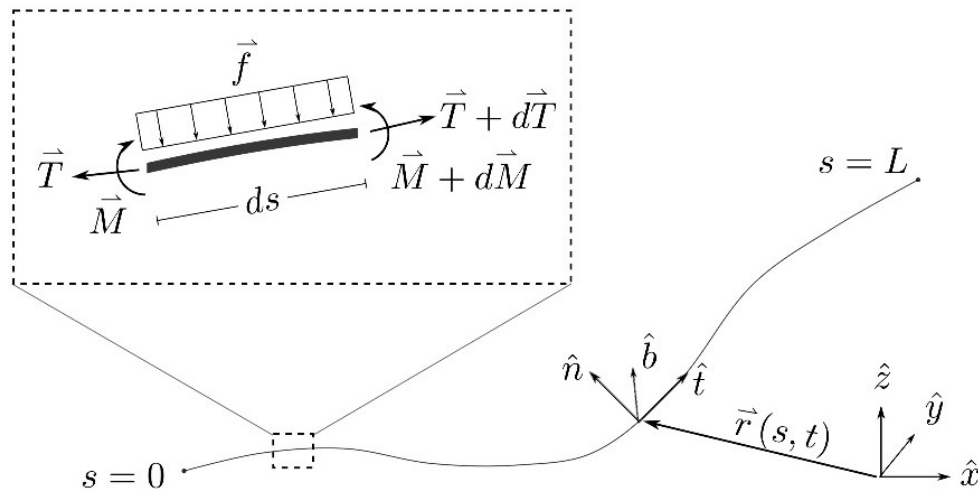
# MOODY FUNCTIONALITY - OVERVIEW

- Cables
- Components
- Rigid bodies
- Hydrodynamic bodies
- Statics
- Environment
- Nemoh interface



# MOODYCORE FUNCTIONALITY – CABLE DYNAMICS

- Parametrisation along unstretched coordinate  $s$
- Added mass and drag from Morison equation
- Ground interaction as spring-damper plane with Coulomb friction damping.
- Bending stiffness implementation adapted from Tjavaras (1999) to a DG formulation
- Formulation is torsion free.



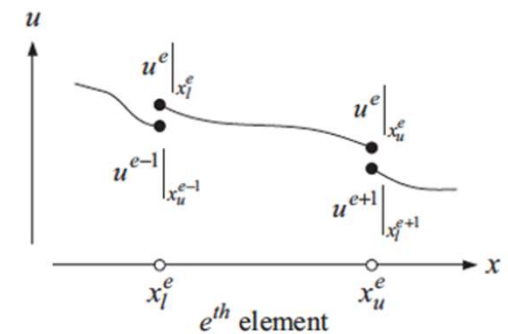
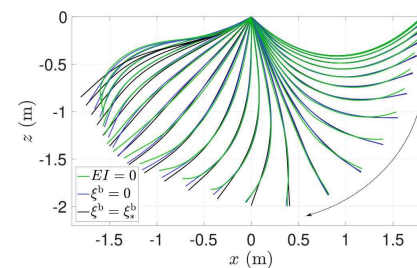
$$\frac{d\vec{v}}{dt} = \frac{d\vec{T}}{ds} + \vec{f}_e,$$

$$\vec{T} = T(\epsilon, \dot{\epsilon})\hat{t} + \vec{T}_s,$$

$$0 = \frac{\partial}{\partial s} \left( \frac{\vec{M}}{l_\epsilon^2} \right) + l_\epsilon \hat{t} \times \vec{T}_s,$$

$$\vec{M} = GI_p \Omega_1 \hat{t} + \hat{t} \times (EI\kappa + \xi_b \dot{\kappa}),$$

Nodal	Modal
$\varphi_0 = 0.5(1 + \xi)$	$\varphi_0 = 1$
$\varphi_1 = 0.5(1 - \xi)$	$\varphi_1 = \xi$



# MOODYCORE FUNCTIONALITY – LINE DYNAMICS

- Line equation is a wave equation
- Hyperbolic solution → Lax Friedrich flux
  - Explicit time stepping
  - Very low numerical damping
  - Limitation when very high internal damping is added.
- Suitable for most ropes and for chains
- Typical dynamic regions of chains:
  - Quasi-static condition [I]
  - Harmonic oscillation [II]
  - Snapping range [III]
  - Free-fall – chaotic and irregular [IV]

$$\frac{d\vec{v}}{dt} = \frac{d\vec{T}}{ds} + \vec{f}_e,$$

$$\vec{T} = T(\epsilon, \dot{\epsilon}) \hat{i} + \vec{T}_s,$$

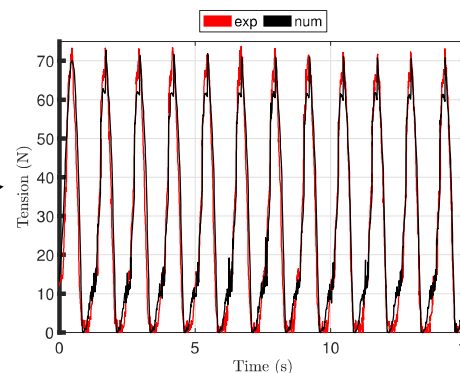
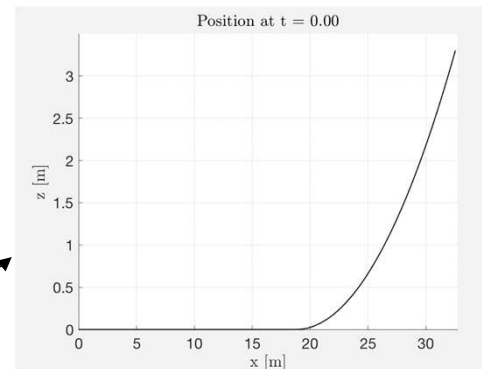


Fig. 1 - Notation diagrams of mooring chain.

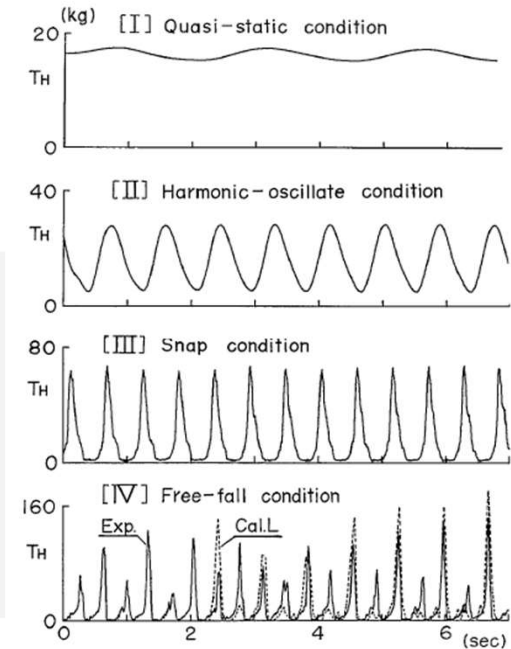
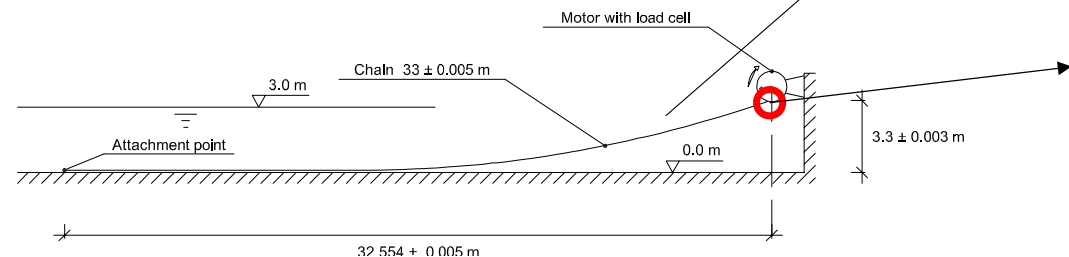


Fig. 2 - Time records of tension of mooring chain.

OTC 4053

DYNAMIC BEHAVIOR AND TENSION OF OSCILLATING MOORING CHAIN

by Toshiro Suhara, Wataru Kotera, Fukuza Tasei, Hironori Miyama, Kyushu University; Kumihiro Sato, Kunito Watanabe, Mitsui Ocean Development & Engineering Co., Ltd.



# MOODYCORE FUNCTIONALITY – CABLES & CABLE TYPES

- Cable type

- Mass/m
- Density of material
- Diameter
- Material model
  - Stiffness, damping, bending etc
- Drag and added mass coefficients

- Cable

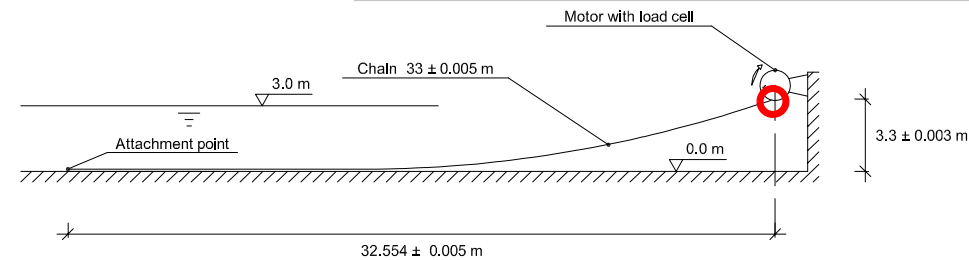
- End point vertex numbers
- Number of elements (P=4<sup>th</sup> order)
- Length
- Initial condition (IC)
  - catenary, straightLine, preStrain...

Input setting combination			
gamma0 (kg/m)	rho (kg/m3)	D(m)	Abuoyant
yes	yes	derived	A(gamma0,rho)
no	yes	required	A(D)
yes	no	required	A(D)
yes	yes	yes	A(rgamma0,rho)

```

%---Cable types---%
cableType1 = {
    diameter = 0.0022;
    rho = 7800;
    gamma0 = 0.0818;
    CDn = 2.5;
    CDt = 0.5;
    CMn = 3.8;
    materialModel = {
        EA = 1e4; % specific input to material model.
        xi = 2;
        type = 'bilinear';
    }
}

%----- Cables -----%
cable1 = {
    typeNumber = 1;
    vertex0 = 1; %
    vertex1 = 2; %
    length = 33; %
    IC.type = 'catenary';
    N = 20; %
}
    
```





# MOODYCORE FUNCTIONALITY - COMPONENTS

- Models axial forces between two points
- Not a dynamic object (massless)
- Input is location and velocity of the two points.
- Output is axial force
- Spring
  - Polynomial stiffness based on extension (m) and axial velocity (m/s)
- Tabulated component
  - Two-dimensional lookup table on total length and extensional velocity
  - Can be used to implement general passive functions
  - End-stops and directional dampers implicitly supported

## Tabulated component

Lookup table of component force (N) based on input velocity (m/s) and length (m)								
	0	39	40	50	60	61	100	Length (m)
-10	-1.04E+04	-10400	-900	-400	100	9600	9600	Saturated damping
-1	-1.04E+04	-10400	-900	-400	100	9600	9600	400Ns/m linear damping
0	-1.00E+04	-1.00E+04	-500	0	500	1.00E+04	1.00E+04	Spring action
1	-9.60E+03	-9600	-100	400	900	10400	10400	400Ns/m linear damping
10	-9.60E+03	-9600	-100	400	900	10400	10400	Saturated damping
Velocity (m/s)								

## Spring component

$$F_c = \sum_{i=1}^N C_i (L - L_0)^i$$

$$F_d = \sum_{i=1}^N D_i v^i$$

**component1**

Specify component properties such as stiffness and mass properties.

**Connectivity**

Start vertex:

End vertex:

**Properties**

Type:

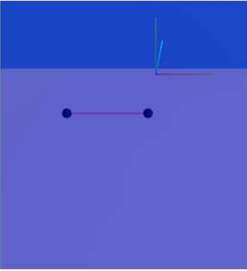
Stiffness (N/m<sup>1</sup>):

Damping (N/s<sup>1</sup>/m<sup>1</sup>):

Rest length (m):

Constant force (N):

Allow compression:

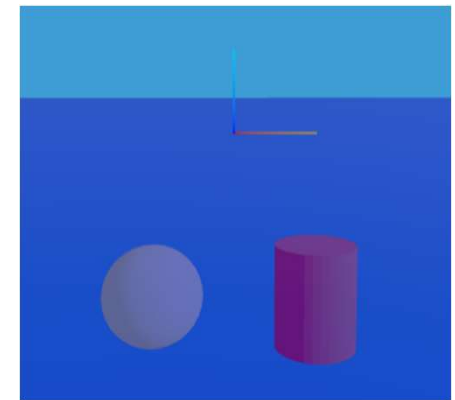


# MOODYCORE FUNCTIONALITY – RIGID BODY

- Dynamic rigid body object. Models inertial properties
  - State is 13 dofs:  $(\vec{r}, \vec{q}, \vec{v}, \vec{\omega})$
  - Position, quaternion, velocity and angular velocity
  - Computed about CoG.
  - Attachment points modelled by slave vertices
  - Output in MoodyMarine shows rotation in Euler angles (degrees)
- Points
  - Points are modelled as spheres.
  - Free-surface interaction is implemented with partial volume.
  - No rotational properties implemented. (3DoF motion)
  - Technically attachments (slaves) work, but moments are not transferred into rotation.
- Cylinder
  - Cylinders were designed to be submerged
  - **NEW IN v3.1:** Free surface interaction is handled approximately, and works best for spar-buoy types, i.e. symmetry axis is vertical
  - Computes full dynamic state of the body.
  - Supports buoyancy offset (i.e. ballasting) along symmetry axis.

**Table 5.2:** Output structure of rigidBodyX.dat. The 14 data columns are time (1), position (3), quaternion (4), velocity (3) and angular velocity (3). The velocity and angular velocity are in the body local coordinate system.

$t_1$	$\vec{p}(t_1)$	$\vec{q}(t_1)$	$\vec{v}(t_1)$	$\vec{w}(t_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$t_m$	$\vec{p}(t_m)$	$\vec{q}(t_m)$	$\vec{v}(t_m)$	$\vec{w}(t_m)$



# MOODYCORE FUNCTIONALITY – HYDROBODY

- **Hydrobody – A wave-body interaction type body**
  - Derived from rigidbody type
  - Includes 1<sup>st</sup> order linear potential flow loads
- **Available type: linearIRF**
  - Impulse response function integration used in radiation forces
- **Linearisation of responses**
  - Small rotations in formulation: Euler angles  $\vec{\theta}$  used.
  - State is 12 dofs:  $(\vec{r}, \vec{\theta}, \vec{v}, \vec{\omega})$
  - Global orientation used for hydrodynamic loads
- **Output**
  - hydroBodyX.dat: Positions and Velocities
  - hydroBodyX\_forces.dat: Hydrodynamic forces and damping forces

$F_{rad}$  Radiation force from the radiation potential  $\Phi_{rad}$ , realised through the radiation Kernel (also known as the impulse response function) convolution integral with body velocity. Linearised at the initial body position.

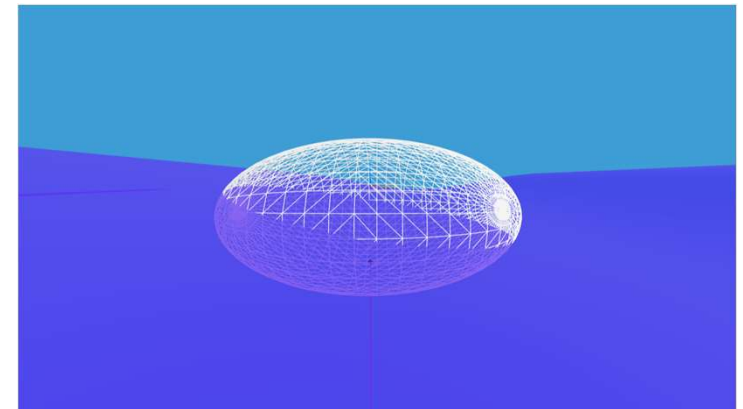
$F_{df}$  Diffraction force from the diffraction potential  $\Phi_{df}$  (sometimes referred to as the scattered potential).

$F_{fk}^{(d)}$  Dynamic Froude-Krylov force from the integral of dynamic pressure in the incident wave potential  $\Phi_I$  over the wetted body surface.

$F_{fk}^{(s)}$  Static Froude-Krylov force, being the hydrostatic pressure integrated over the wetted body surface.

$F_{fk}$  Total Froude-Krylov force, the sum of static and dynamic Froude-Krylov forces  $F_{fk} = F_{fk}^{(s)} + F_{fk}^{(d)}$ .

$F_{ex}$  Complex-valued excitation force,  $F_{ex} = F_{fk}^{(d)} + F_{df}$



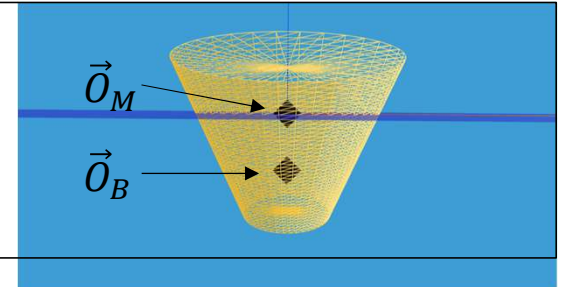
# MOODYCORE FUNCTIONALITY –HYDROBODY

- Mesh positioning -- Reballasting

- Vertex is at body CoG. ( $\vec{O}_B$ )
- Mesh is allowed a different origin ( $\vec{O}_M$ )
- CoG in mesh: Position of CoG in MESH coordinates, i.e.  $\vec{O}_B - \vec{O}_M$ .
- Internally, moodyCore shifts the mesh coordinate so that its origin is at CoG.
- Example of workflow scenario seen to the right

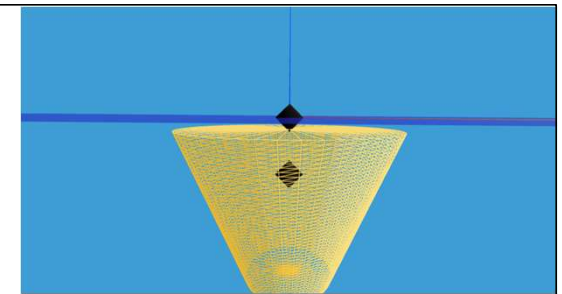
## STEP 1 (IMPORT MESH)

$V1=(0,0,0) = \vec{O}_M$   
 $V2=(0,0,-0.1) = \vec{O}_B = \text{CoG}$   
Body vertex=1  
CoG in mesh=(0,0,0)



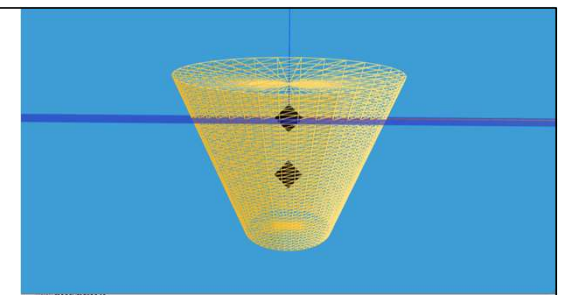
## STEP 2 (SELECT VERTEX)

Body vertex=2  
CoG in mesh=(0,0,0)  
Mesh moves to new point



## STEP 3 (SET COG IN MESH)

CoG in mesh=(0,0,-0.1)  
Mesh moves UP 0.1m



# MOODYCORE FUNCTIONALITY –HYDROBODY

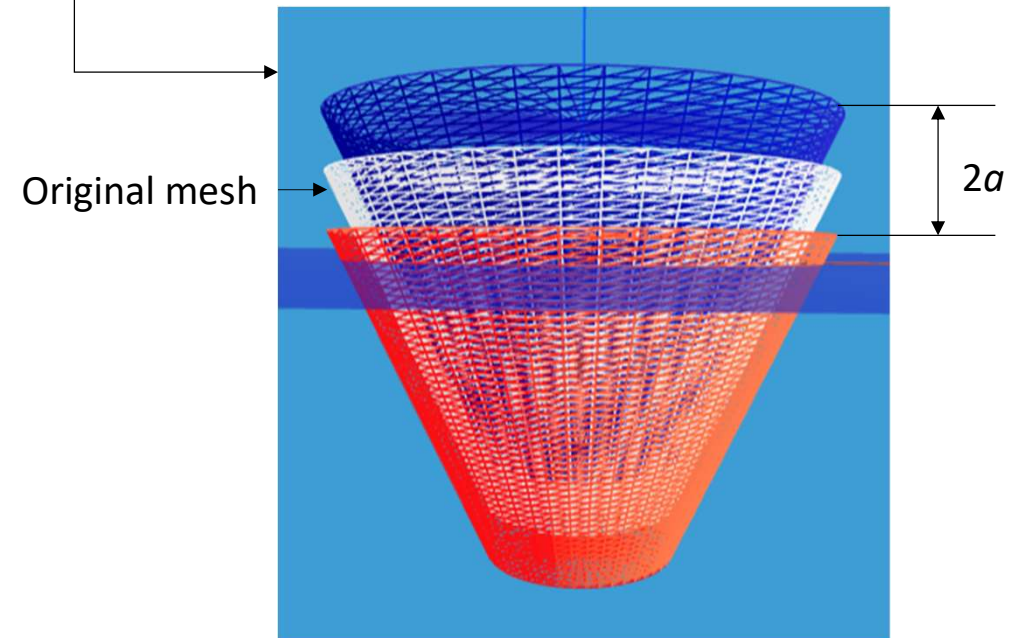
- Mesh positioning -- Reballasting

- Vertex is at body CoG. ( $\vec{O}_B$ )
- Mesh is allowed a different origin ( $\vec{O}_M$ )
- CoG in mesh: Position of CoG in MESH coordinates, i.e.  $\vec{O}_B - \vec{O}_M$ .
- Internally, moodyCore shifts the mesh coordinate so that its origin is at CoG.

- Buoyancy modes

- **Linear-matrix**  
A linear C-matrix is provided (heave only, surge-heave-pitch, or full matrix)
- **Linear-derived**  
Linear stiffness is derived from the mesh disposition, based on a secant method linearization over an amplitude  $a$ .  
Mesh cutting at the SWL is used at each position
- **Nonlinear**  
The nonlinear Froude-Krylov force and static wave pressure are jointly added to the restoring force.

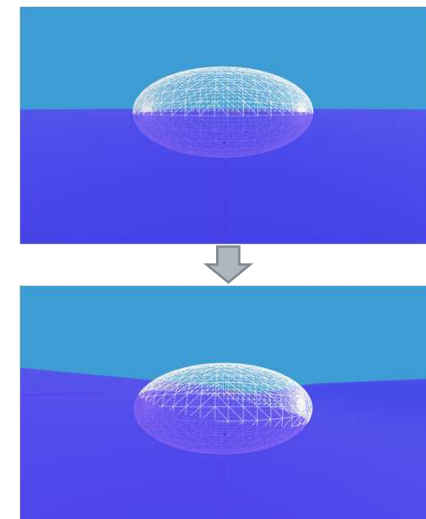
Buoyancy	Input setting combination			
Mode	nlfk	C	volume	meshName
Linear-matrix	0	found	required	unused
Linear-derived	0	not-found	unused	required
Nonlinear	1	unused	unused	required



# MOODYCORE FUNCTIONALITY –HYDROBODY

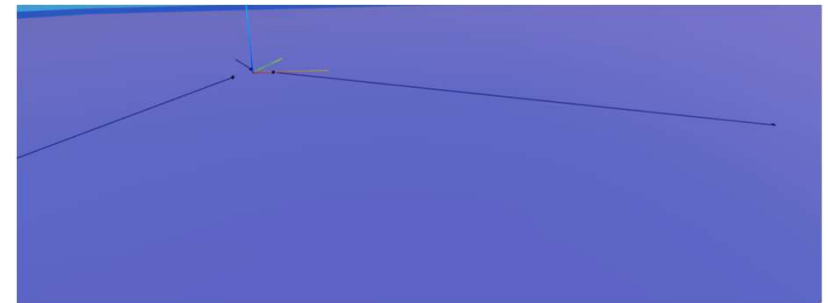
- Mesh positioning -- Reballasting
- Buoyancy modes
- NFLK effects
  - **BEM data**  
To use the nonlinear Froude-Krylov formulation fully, the bem software has to provide the diffraction force separated from the Froude-Krylov excitation.
  - **Output**  
Fexc in output forces are then really Fdiffraction only
  - **Nonlinear**  
The nonlinear Froude-Krylov force and static wave pressure are jointly added to the restoring force.

Buoyancy	Input/Output combination		
Mode	nlfk	Fexc	Frest
Linear-matrix	0	Excitation	C*x
Linear-derived	0	Excitation	C*x
Nonlinear	1	Diffraction	FK+HS



# MOODYCORE FUNCTIONALITY –STATICS

- Static system solver – still experimental
  - Not really static solver per se
  - Relaxed dynamic solution
  - No hard convergence check (yet...)
- Solve stage
  - Rigid bodies and hydro bodies move
  - Cables are solved for using IC analytic conditions
  - Cables are set as quasi-static during solve stage
  - A larger time step can often be suffered here
  - Typically used for floating body draft and hybrid mooring legs for initial buoy/weight position
    - Helps to find position of vertices
- Relax stage
  - Full system dynamics is run
  - Requires many steps with small dt (0.1\*CFL is default)
    - Helps to mitigate high-frequency, transient spikes in the tension of cables during startup



```
statics = {  
  solve= 1;  
  maxIter = 10000;  
  timeStep = 0.1;  
  relax =1;  
}
```



# MOODYCORE FUNCTIONALITY – BEM INTERFACE

- Nemoh
  - Solves the linear potential flow force coefficients for a floating body
  - Developed at LHEEA Centrale Nantes
- MoodyCore interface
  - Nemoh project writer through moodyPre.exe
  - Includes mesh IO (.dat, .stl, .gdf)
  - Nemoh output reader into moodyCore.
  - Hydrostatic results not required
- Capytaine reader experimentally implemented
  - To be released when tested with latest functionality

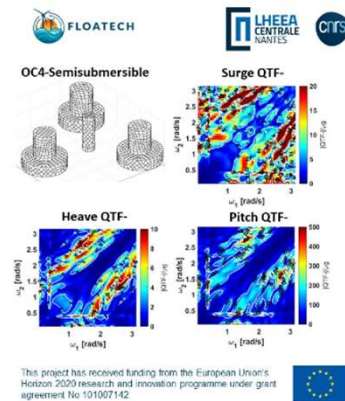
## NEMOH-Presentation

Discover all the information about NEMOH via the menu on the right

NEMOH is a Boundary Element Methods (BEM) code dedicated to the computation of first order wave loads on offshore structures (added mass, radiation damping, diffraction forces). It has been developed by researchers at Ecole Centrale de Nantes for 30 years. It is still used in many of our research projects. Typical use is estimation of dynamic response of floating structures or performance assessment of wave energy converters.

Unlike other BEM softwares, NEMOH's approach decouples the resolution of the linear free surface Boundary Value Problem (BVP) and the definition of the boundary condition on the body (body condition). This feature makes it easy to deal with flexible structure, hydroelasticity, generalised modes and unconventional degrees of freedom with NEMOH.

NEMOH is the world's first open source BEM code. The release NEMOH v3.0.0 includes several recent developments. The first-order module allows for removing irregular frequencies and new solvers are available for the linear system, resulting in enhanced computational efficiency. A new extension module is provided for computing quadratic transfer functions (QTFs). This release will be, to our knowledge, the sole and only open-source BEM software that provides the second-order module. The software is released under the GNU General public v3 license.





# MOODYCORE FUNCTIONALITY - ENVIRONMENT

- Regular waves (airy)
- JONSWAP spectrum
- Custom wave input
  - Can be used to model wave trains in different directions
- Current
  - Constant, unidirectional, depth-dependent
- Wind
  - Constant in time, optional power-law in height
- Ground model
  - Spring-damper horizontal plane
  - Horizontal friction as Coulomb damping

```
wave = {  
    type = 'regular';  
    amplitude = 1;  
    period = 6;  
    phase = 0;  
    rampTime = 30;  
    depth = 75;  
}
```

$$\eta = a \cos(k_x x + k_y y - \omega t + \phi)$$

```
%--- Ground model input -----%  
ground = {  
    level = 0;  
    type = 'springDamp';  
    % damping = 1e4;  
    dampingCoeff = 1; % ratio of critical damping for each cable  
    frictionCoeff = 0.3;  
    frictionVelocity = 0.01;  
    stiffness = 3e9;  
    oneWayDamping = 0;  
}
```

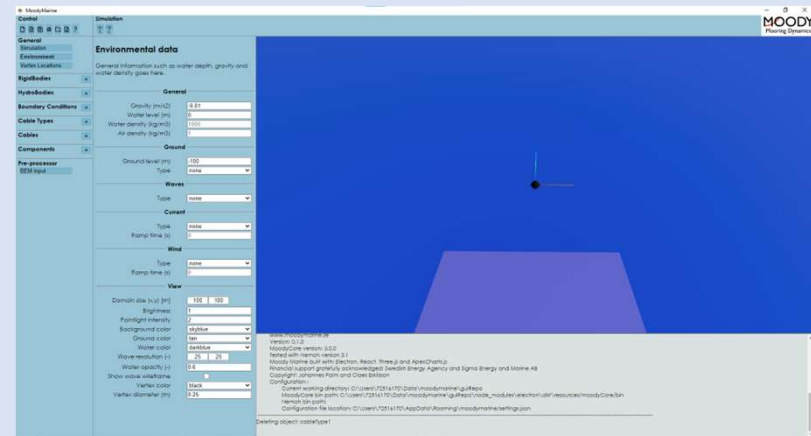
$$f_c^{(xy)} = \sqrt{l_\epsilon} \mu \tanh\left(\pi \frac{|\vec{v}_{xy}|}{v_\mu}\right) \frac{\vec{v}_{xy}}{|\vec{v}_{xy}|} \min(f_b^{(z)}, 0),$$

$$f_c^{(z)} = \sqrt{l_\epsilon} \left( K_g d \delta - \xi 2 \sqrt{K_g d \gamma_0 v_z} \right),$$

# MOODYCORE FUNCTIONALITY - USAGE

- ✓ Cables
- ✓ Components
- ✓ Rigid bodies
- ✓ Hydrodynamic bodies
- ✓ Statics
- ✓ Environment
- ✓ Nemoh interaction
- **Using it**

## LIVE DEMO



# USAGE – INSTALLATION & TUTORIALS

- MoodyCore

- moody.exe
- moodyPre.exe
- moodyPost.exe
- libMoodyWrapper (.so, .dylib,.dll)
- moodyWrapper.h

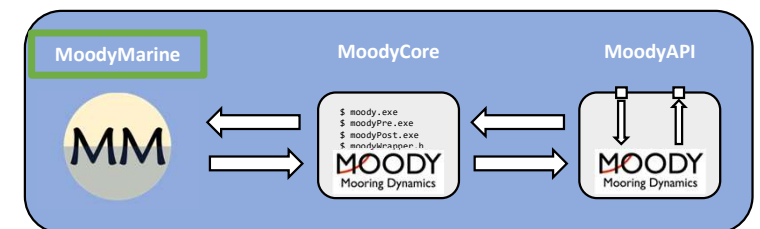
- Installation

- **moodyMarine:** Run the installer for your OS (win,linux,mac)
- Check installation by opening and run statics in the empty case or with a default rigid body point added.
- **moodyCore:** Download and unpack tar/zip archives  
For linux and mac: look in etc for environmental variables.  
Windows: add to path in env. variable
- **Note:** MoodyMarine is self-contained. You do not need moodyCore to use it.
- MoodyCore is needed for advanced usage through the command-line interface, as well as for MoodyAPI usage.

- MoodyMarine

- Tutorials

- T1 – Single catenary
- T2 – Multiple cables
- T3 – Swinging cable
- T4 – Uppsala generator mockup
- T5 – Submerged buoys
- T6 – Ellipsoid with damper
- T7 – Hywind 5MW platform



# USAGE – INSTALLATION & TUTORIALS

- MoodyCore

- moody.exe
- moodyPre.exe
- moodyPost.exe
- libMoodyWrapper (.so, .dylib,.dll)
- moodyWrapper.h



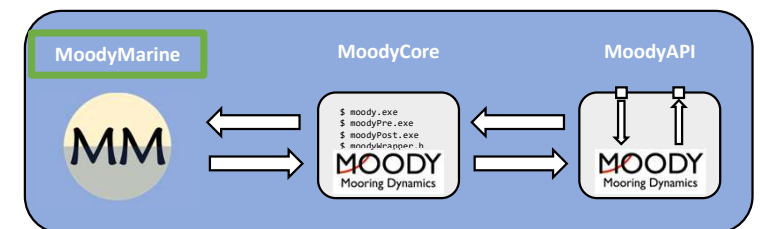
- Installation

- **moodyMarine:** Run the installer for your OS (win,linux,mac)
- Check installation by opening and run statics in the empty case or with a default rigid body point added.
- **moodyCore:** Download and unpack tar/zip archives  
For linux and mac: look in etc for environmental variables.  
Windows: add to path in env. variable
- **Note:** MoodyMarine is self-contained. You do not need moodyCore to use it.
- MoodyCore is needed for advanced usage through the command-line interface, as well as for MoodyAPI usage.

- MoodyMarine

- Tutorials

- T1 – Single catenary
- T2 – Multiple cables
- T3 – Swinging cable
- T4 – Uppsala generator mockup
- T5 – Submerged buoys
- T6 – Ellipsoid with damper
- T7 – Hywind 5MW platform



# USAGE – TUTORIAL 1: SINGLE CATENARY

1. Vertex locations
2. Environment (ground)
3. Cable properties
4. Boundary conditions
5. Compute
6. View results

## MoodyCore Command line

```
PS C:\Users\72516170> moody.exe -f lindahl125.m
```

```
%--- Simulation settings ---%
simulation = {
  time = {
    start = 0; % [s] Start time
    end = 16; % [s] End time of simulation
    dt = 1e-4; % [s] Time step size
    % cfl = 0.9; % use adaptive time step based on max courant number
  }
  print = {
    dt = 1e-2; % save results every xx time interval
    % format = "ascii"
  }
}

%--- Environmental settings ---%
environment = {
  gravity = -9.81; % [m/s2] Gravitational acceleration
  waterLevel = 3; % [m] z-coordinate of mean water level
  waterDensity = 1e3; % [kg/m3] Density of water
  airDensity = 0.0; % [kg/m3] Density of air

  %--- Ground model input ---%
  ground = {
    level = 0;
    type = 'springDamp';
    % damping = 1e4;
    dampingCoeff = 1; % ratio of critical damping for each cable
    frictionCoeff = 0.3;
    frictionVelocity = 0.01;
    stiffness = 3e9;
    oneWayDamping = 0;
  }
} % end environment

%--- List of vertices ---%
%
vertexLocations = {
  {
    no. x y z
    1 [ 0 0 0 ];
    2 [ 32.554 0 3.3 ]
  };
};
```

```
%---Cable types---%
cableType1 = {
  diameter = 0.0022;
  rho = 7800;
  gamma0 = 0.0818;
  CDn = 2.5;
  CDt = 0.5;
  CMn = 3.8;
  materialModel = {
    EA = 1e4;
    xi = 2;
    type = 'bilinear';
  }
}

%---- Cables ----%
cable1 = {
  typeNumber = 1;
  vertex0 = 1; %
  vertex1 = 2; %
  length = 33; %
  IC.type = 'catenary';
  N = 20; %
}

%--- Boundary Conditions ---%
bc1 = {
  vertex = 1;
}
bc2 = {
  vertex = 2;
  mode = "sine";
  rotationMode = "pinned";
  % if scalar values, it is applied to all dimensions:
  amplitude = [0.2;0;-0.2]; % [m]
  frequency = [0.8;0.8;0.8]; % [Hz]
  phase = [90;0;0]; % [deg]
  centerValue = [32.554;0;3.3]; % [m]
  rampTime = 2.5; % [s]
}
```

# USAGE– IO AND FILE FORMATS

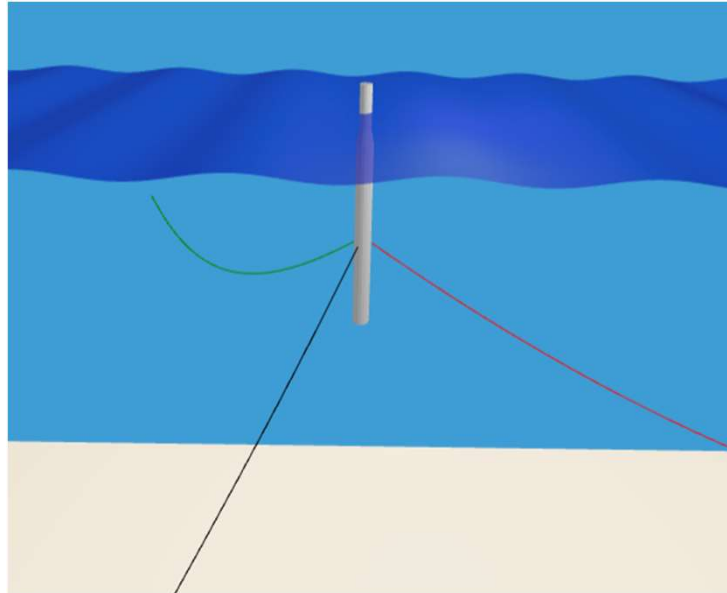
- **MoodyCore**
  - Reads matlab-like text file OR .json format.
  - .m extension is only to view in matlab. Can be anything
  - .json is required for .json format though.
  - Based on custom readers implemented in c++
- **Simulation data**
  - -o flag specifies output name of simulation when missing *inputfile.m* results are created as *inputfile*
  - Use convention to place result folders in the same folder as the input file to enable relative paths in input file (to timeseries, meshes etc)
  - The results of a simulation contain setup.json and setup.txt for compatibility
  - Static and dynamic results
- **MoodyMarine**
  - Reads .json data file format
  - On simulation, we execute moodyCore using a saved input file. Therefore, MoodyMarine prompts you to save the input file prior to running the static or dynamic simulation.
  - Can open moodyCore results (by reading the setup.json and info.json files)
  - A direct converter tool of input formats is pending development
- **Mesh manipulation**
  - moodyPre.exe can be used to modify meshes (translate origin, location, cut at the waterline, and convert between .stl, .dat (nemoh) and .gdf(WAMIT) formats).
  - It is used by MoodyMarine to enable input meshes to be other than .stl formats
  - A <meshname>\_wetted.dat file is written to the Nemoh project upon executing the Nemoh integration part of MoodyMarine.

```
wave = {  
  type = 'regular';  
  amplitude = 1;  
  period = 6;  
  phase = 0;  
  rampTime = 30;  
  depth = 75;  
}
```

```
"wave": {  
  "type": "regular",  
  "amplitude": 2,  
  "period": 7,  
  "phase": 0,  
  "direction": 0,  
  "rampTime": 21,  
  "none": {},  
  "regular": {  
    "amplitude": 2,  
    "period": 7,  
    "phase": 0,  
    "direction": 0,  
    "rampTime": 21  
  },  
}
```

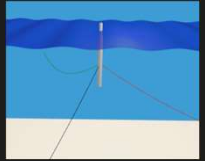
# USAGE – TUTORIAL 7: HYWIND SPAR PLATFORM

- 1) Prepare body
  - Load mesh
  - Set CoG and inertial properties
  - Set constraints (opt)
- 2) Vertex locations. Input
- 3) Set cable properties
- 4) Set environment
- 5) Setup BEM preprocessor
- 6) Simulate results
- 7) View results



```
hydroBody1 = {  
  vertex = 1;  
  slaves = [  
    3 [5.2 0 8.01] 0 [0 0 0]  
    5 [-2.6 4.5033 8.01] 0 [0 0 0]  
    7 [-2.6 -4.5033 8.01] 0 [0 0 0]  
  ];  
  mass = 8066048;  
  constraints = [];  
  I = [4229230000,4229230000,164230000];  
  meshName = "hywind_20mTower.stl";  
  cogInMesh = [0,0,-78.01],  
  nlfk = 0;  
  hydroData = "./nemoh";  
  type = linearIRF;  
  IRF = {  
    time = 30;  
    dt = 0.01;  
    save = 1  
  };  
  view = {  
    color = "grey";  
    wireframe = 1;  
  }  
}
```

# USAGE – TUTORIAL 7: HYWIND SPAR PLATFORM



- 1) Prepare body
- 2) Vertex locations. Input
  - Only free vertices required
  - Anchor bcs defined
  - Fairlead positions defined in each body as slaves
    - NOTE: Vertices that are slaves are displayed as is in input mode. BUT the values in slave input has precedence in solution. After statics, slave data is seen.
- 3) Set cable properties
- 4) Set environment
- 5) Setup BEM preprocessor
- 6) Simulate results
- 7) View results

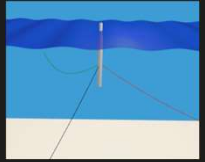
```
vertexLocations = [  
  1  0      0      -78.01  
  2 853.87  0      -320  
  3  5.2    0      -70  
  4 -426.935 739.4731 -320  
  5  -2.6   4.5033  -70  
  6 -426.935 -739.4731 -320  
  7  -2.6   -4.5033  -70  
]
```

```
%% --- BCs are all anchor points  
bc1.vertex = 2;  
bc2.vertex = 4;  
bc3.vertex = 6;
```

```
hydroBody1 = {  
  vertex = 1;  
  slaves = [  
    3 [5.2 0 8.01] 0 [0 0 0]  
    5 [-2.6 4.5033 8.01] 0 [0 0 0]  
    7 [-2.6 -4.5033 8.01] 0 [0 0 0]  
  ];  
  mass = 8066048;  
  constraints = [];  
  I = [4229230000,4229230000,164230000];  
  meshName = "hywind_20mTower.stl";  
  cogInMesh = [0,0,-78.01],  
  nlfk = 0;  
  hydroData = "./nemoh";  
  type = linearIRF;  
  IRF = {  
    time = 30;  
    dt = 0.01;  
    save = 1  
  };  
  view = {  
    color = "grey";  
    wireframe = 1;  
  }  
}
```



# USAGE – TUTORIAL 7: HYWIND SPAR PLATFORM



- 1) Prepare body
- 2) Vertex locations. Input
- 3) Set cable properties
  - Length, stiffness,
  - Damper values
- 4) Set environment
  - Ground
  - Waves
- 5) Setup BEM preprocessor
- 6) Simulate results
- 7) View results

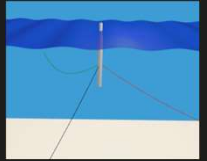
```
%% --- Environmental settings
environment = {
    gravity= -9.81;
    waterLevel= 0;
    waterDensity= 1025

    % Set the ground
    ground = {
        level= -320
        type = "springDamp"
        stiffness = 1000000
        dampingCoeff = 1
        damping = 1000
        frictionCoeff = 0.1
        frictionVelocity = 0.05
    };
    % Make waves
    wave = {
        type = "regular"
        amplitude = 2
        period = 7
        phase = 0
        direction = 0
        rampTime = 20
    };
};
```

```
%% --- Cable types
cableType1 = {
    gamma0 = 77.7066;
    diameter = 0.09;
    rho = 12170;
    CDn = 1.875;
    CDt = 0.86;
    CMn = 0.5;
    CMt = 0;
    materialModel = {
        EA = 384243000;
        xi = 500;
        type = "bilinear";
    }
};

%% --- Cables
cable1 = {
    vertex0 = 2;
    vertex1 = 3;
    type = 1;
    length = 902.2;
    N = 10;
    IC.type = "catenary";
    view = {
        color = red;
    };
};
```

# USAGE – TUTORIAL 7: HYWIND SPAR PLATFORM



- 1) Prepare body
- 2) Vertex locations. Input
- 3) Set cable properties
  - Length, stiffness,
  - Damper values
- 4) Set environment
- 5) Setup BEM preprocessor
- 6) Simulate results
- 7) View results

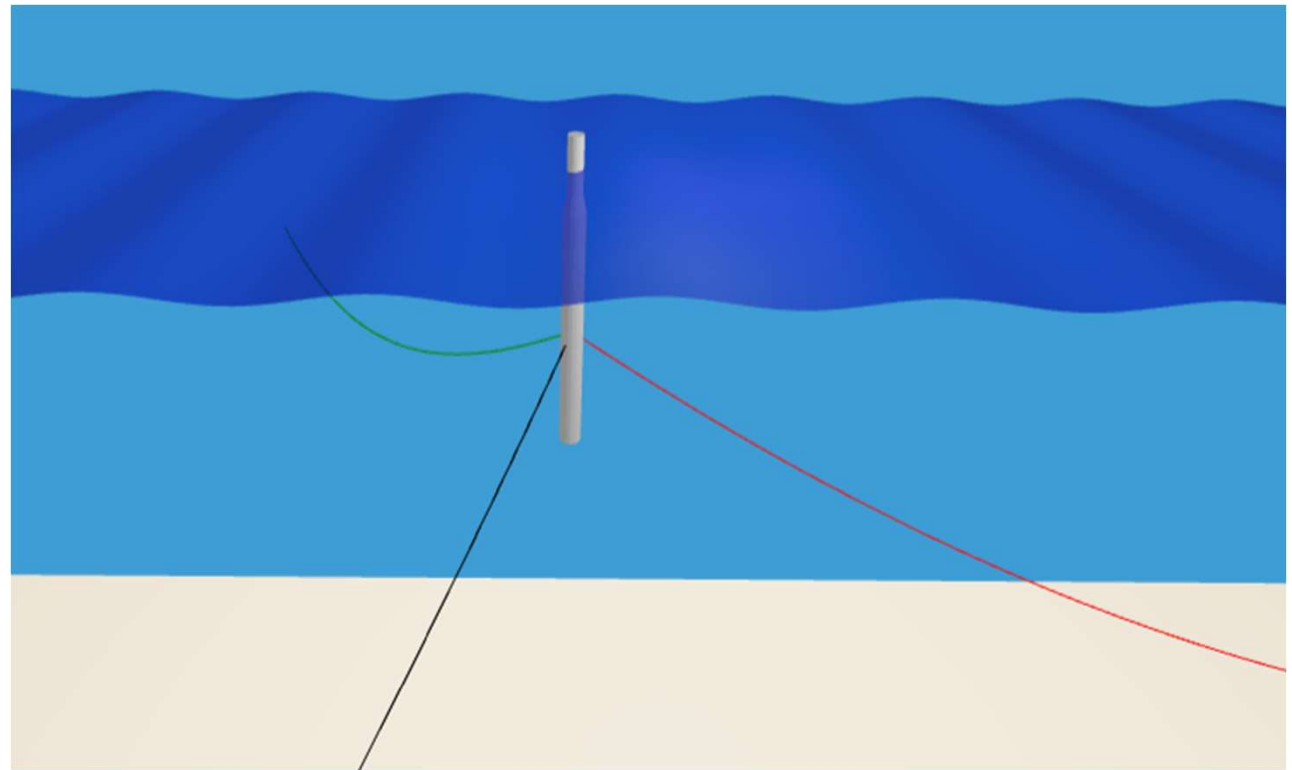
```
%% --- NEMOH INPUT BELOW --- %%
% Info below is only used by moodyPre.exe
% moodyPre.exe -nemoh -f <thisInputFile> generates a Nemoh project.
bem = {
    body = 1;
    type="nemoh"
    output = "nemoh";
    w0 = 0.0628;
    w1 = 4.082;
    nFreqs = 35;
    dir0 = 0;
    dir1 = 0;
    nDirs = 1;
};

% Body specific nemoh-input (used by moodyPre.exe -nemoh)
hydroBody1.bemInput = {
    meshName = "hywind_20mTower.stl";
    keepMesh = 0;

    % CoG of tower, nacelle and SPAR platform. in mesh coordinates (mesh origin at SWL)
    cogInMesh = [0, 0, -78.01];
    position = [ 0, 0, -84.165] % approximate static equilibriae of CoG
};
```

# USAGE – TUTORIAL 7: HYWIND SPAR PLATFORM

- 1) Prepare body
- 2) Vertex locations. Input
- 3) Set cable properties
- 4) Set environment
- 5) Run BEM preprocessor
- 6) Simulate results
- 7) View results
  - Time history
  - Wave animation



NB: Please note that when BEM-data is available, a new <inputName>\_bem folder is created. It contains precomputed impulse-response-functions for the simulation, so that multiple simulations can be run without paying the cost of initial IRF creation.

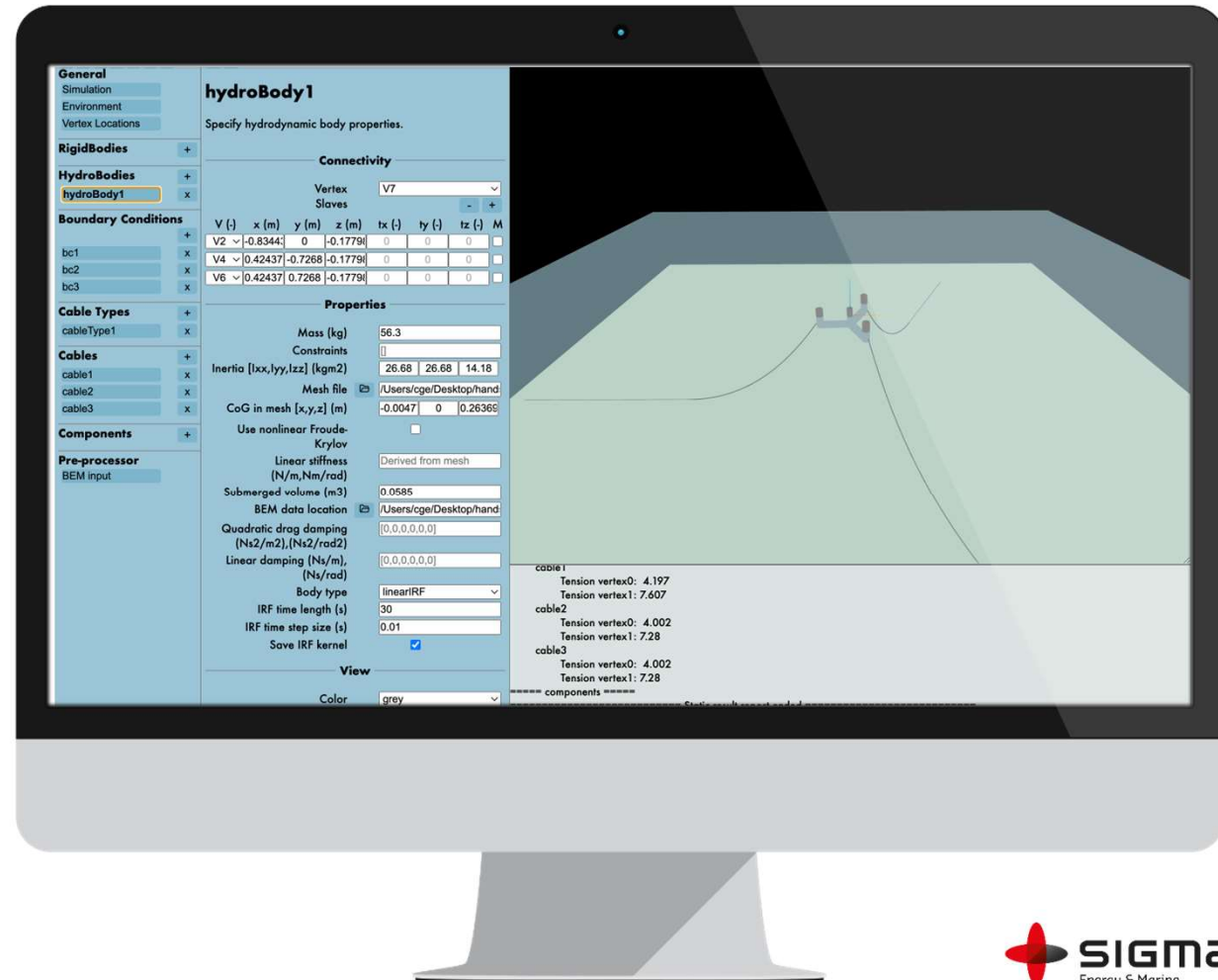
# USAGE – HANDS ON: UMAIN VOTLURNUS-S

Umaine Volturnus-S  
CCP-WSI test case 15

All geometry, mooring, etc data can be  
found at:

[https://www.ccp-wsi.ac.uk/data\\_repository/test\\_cases/test\\_case\\_015](https://www.ccp-wsi.ac.uk/data_repository/test_cases/test_case_015)

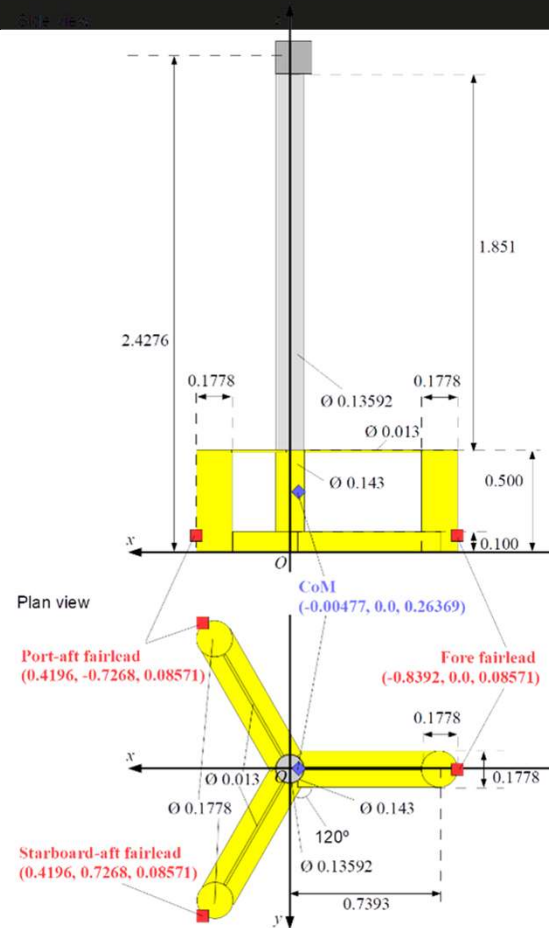
nemohData, stl files and a baseline input  
file are provided and can be downloaded  
from the same folder as the MoodyMarine  
software under folder “handsOn”



# USAGE – HANDS ON: UMAIN VOTLURNUS-S

Open the pre-configured input file. We will go through the steps inside the body required for this case

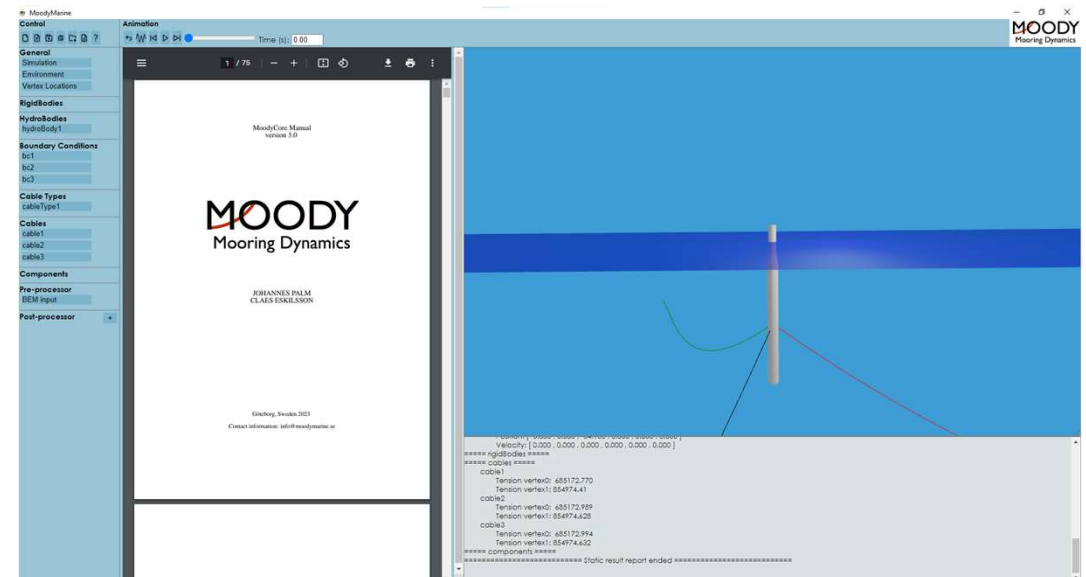
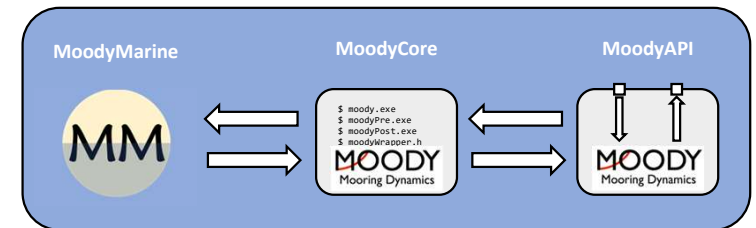
1. Load mesh and set body mass properties  
The stl has body coordinates like the figure on the right  
Shift mesh by cogInMesh property to assign CoG.
2. Continue with adding the vertices: remember fairleads (slaves) are given relative the CoG of the body
3. Create and connect the cables
4. Set the BCs
5. Run with a regular wave  
(start with  $a=0.1\text{m}$ ,  $T=1.25\text{s}$ )
6. Visualize it with waves



NOTE: All dimensions in metres

# FINAL NOTES ON MOODY MARINE

- Wraps moodyCore functionality in a GUI
  - Official release at [moodymarine.se](http://moodymarine.se)
  - Soon out of Beta version --> child sicknesses still to be expected
  - Help found in: Tooltip, Manual and Tutorials
  - Suspected bugs, feature- and help requests: [info@moodymarine.se](mailto:info@moodymarine.se)
- Use it for:
  - Detailed mooring analysis and snap loads
  - Nonlinear FK simulations
  - Simple preprocessor to CFD; static design and dynamic estimates
- Known issues
  - Changing between post-processing results resets zoom level
  - ... to be continued 😊



# OUTLOOK: THANK YOU FOR ATTENDING THIS FAR

- **MoodyCore development plan**
  - Reduce time update requirement on wave dynamics (very expensive to do at each cable step)
  - Multibody interaction in BEM analysis
  - Constrained relative motion between bodies
  - PTO and control implementations
  - 2<sup>nd</sup> order drift forces from BEM analysis (QTFs)
- **MoodyMarine wish list**
  - Convenience functionality (file format robustness, lessons learned from today, usability)
  - Export animation
  - More output data in the plot functions
  - Bug fixes, stability improvements, error messages...
- **MoodyAPI**
  - Packaging developments to the OF rigid body solver
  - Currently, only mooring restraint is included in official release. Older OF version code available at [github.com/johannep/moodyAPI](https://github.com/johannep/moodyAPI)

