MooDy: a dynamic mooring library used for station-keeping simulations in OpenFOAM

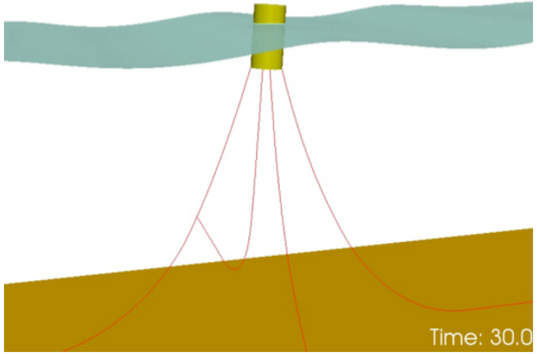
Johannes Palm, Claes Eskilsson and Lars Bergdahl

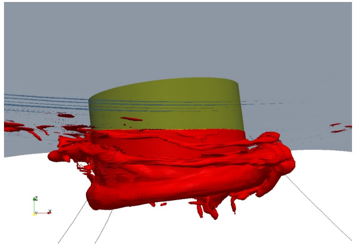
Associate Professor
Dep of Civil Engineering
Aalborg University, Denmark
cge@civil.aau.dk

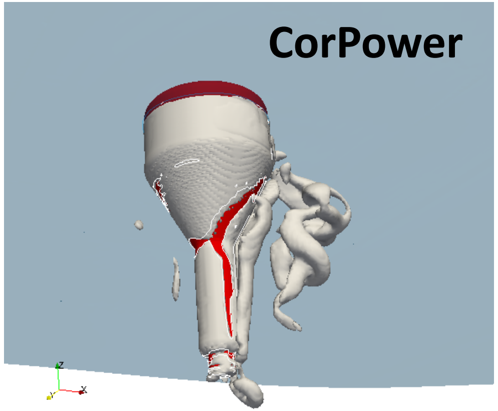
Senior Researcher
Safety and Transport
RISE Research Institutes of Sweden
claes.eskilsson@ri.se

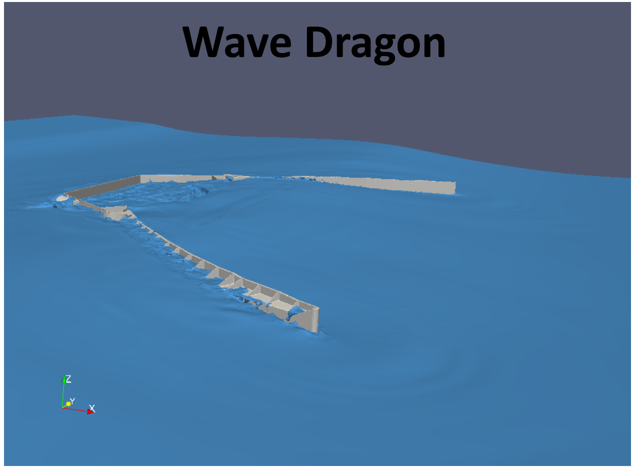
Floating wave energy converters…

Simple forms

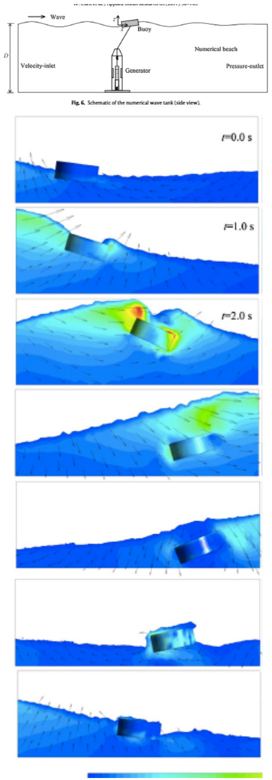






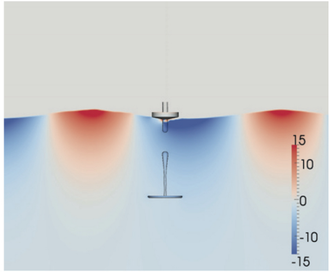


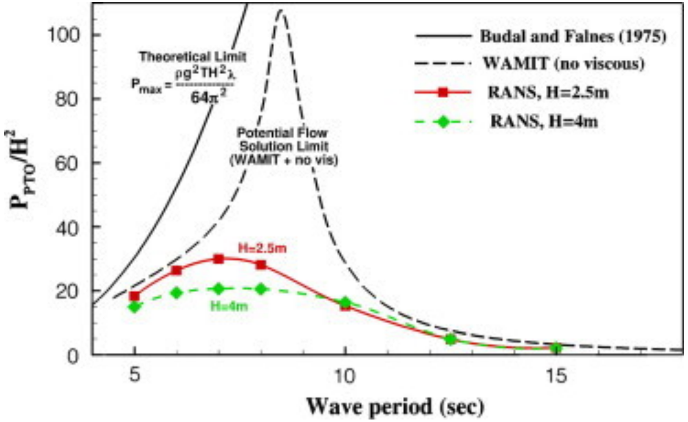
Survival



Chen et al 2016

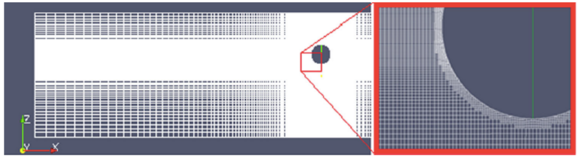
Resonance

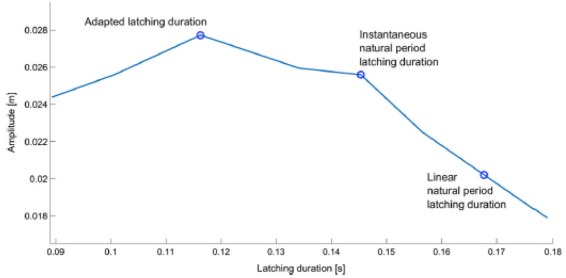




Yu and Li 2013

Control



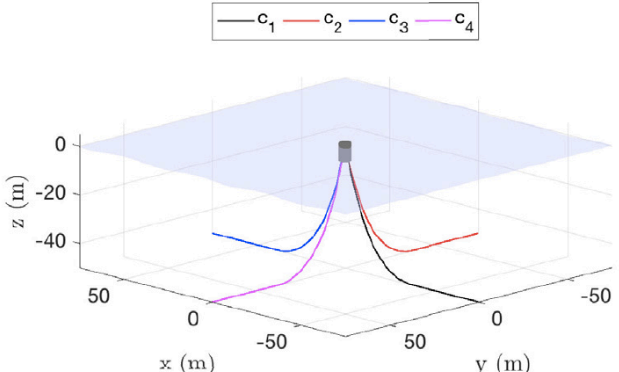


Girogi Ringwood 2016

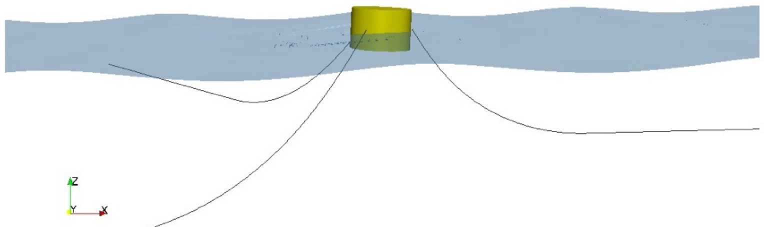
MooDy is a solver for mooring cable dynamics

- Designed to handle snap loads
  - Equations for re-written in conservative form
  - High-order discontinuous Galerkin method
  - *hp*-adaptive solution
  - Explicit time-steppers
- Cables with neglible bending stiffness
- Written in C++ with no dependencies (except std lib)
- Coupled to OpenFOAM and WEC-SIM/FAST
- Released as pre-compiled lib

WEC-SIM (and FAST)



OpenFOAM



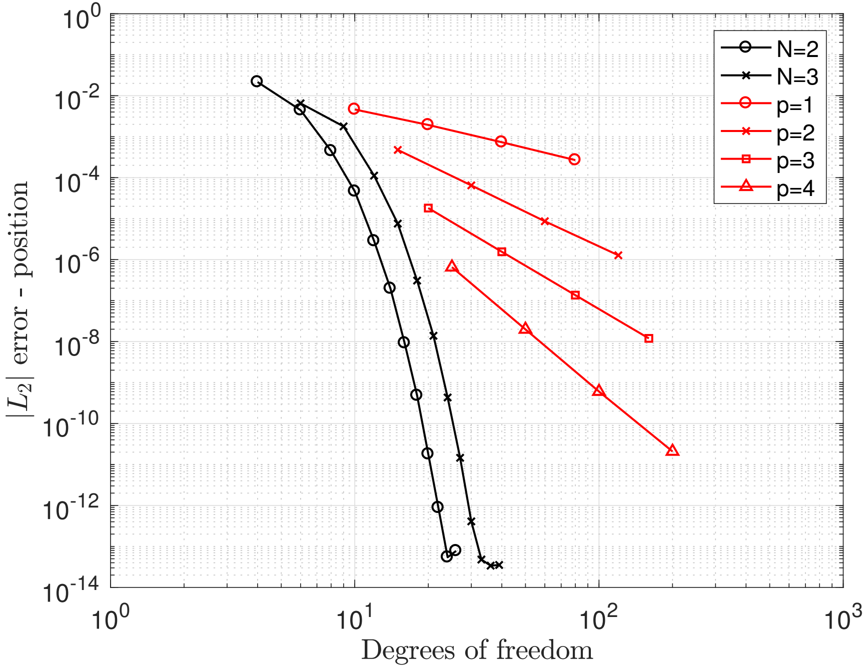
- **Smooth solutions**
  - *Verified on linearised standing wave*
  - *Exponential convergence: h $^{p+1/2}$*

Use high-order elements!

Table 1: Convergence rates for the linear standing wave for position. α is the convergence rate based on $\varepsilon = Ch^{\alpha}$.

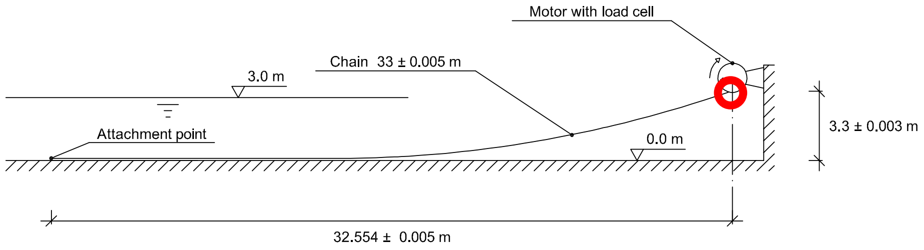
p	ε_{N10}	ε_{N20}	ε_{N40}	$\alpha_{20/10}$	$\alpha_{40/20}$
1	2.13×10^{-3}	7.65×10^{-4}	2.73×10^{-4}	1.48	1.48
2	1.03×10^{-4}	1.30×10^{-5}	1.63×10^{-6}	3.00	2.99
3	1.57×10^{-6}	1.36×10^{-7}	1.20×10^{-8}	3.52	3.51
4	3.75×10^{-8}	1.08×10^{-9}	3.05×10^{-11}	5.12	5.14

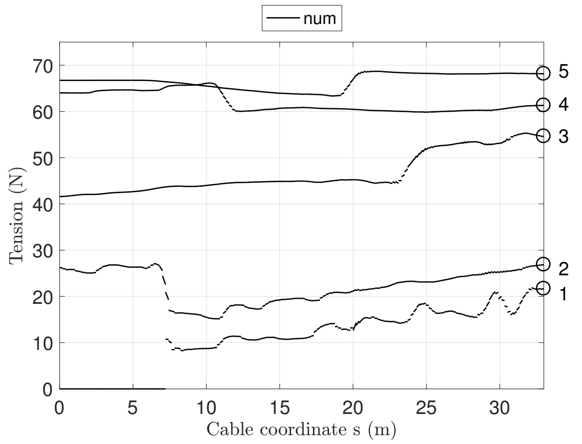
h: Elemental resolution
p: Polynomial order of basis

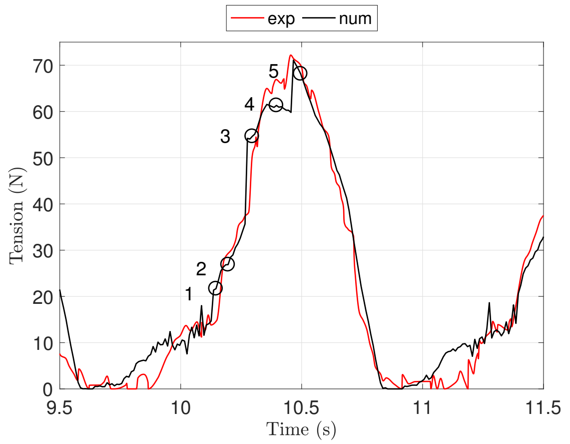


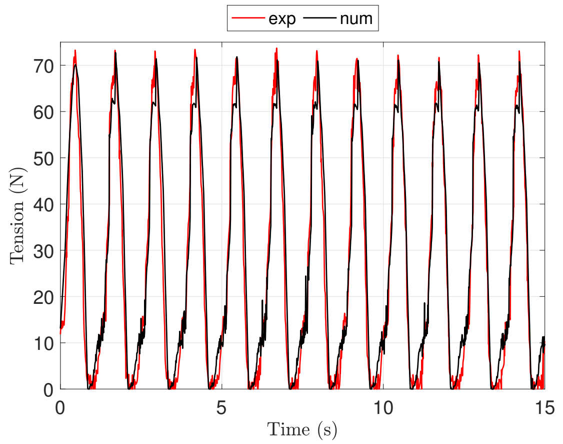
Validation with snap loads

- **Validation case**
  - *Excellent match with experimental data*
  - *Snap load from cable slack*









Station-keeping approaches used

- Heave only → sixDoFRigidBodyMotionConstraints:line/axis
- Linear spring → sixDoFRigidBodyMotionRestraints::linearSpring
- Quasi-static approach → sixDoFRigidBodyMotionRestraints::mooringLine (waves2Foam)
- **Dynamic approach → sixDoFRigidBodyMotionRestraints::moodyR**

```
/*---------------------------------------------------------------------------*\
                      Class moodyR Declaration
\*---------------------------------------------------------------------------*/

class moodyR
:
    public sixDoFRigidBodyMotionRestraint
{
        // Private data

        // Reference points of attachment to the solid body //
        List<point> attachPts_;

        // Word containing the filename of the input file for the
mooring system //
        string mooringSystemFilename_;

        // The number of attachment points //
        int noOfAttachments_;

        // The ramp time of the mooring force (not invoked in moody,
but before force is returned to motion_ ) //
        scalar mooringRampTime_;

        // Last call time of function //
        scalar startTime_;

        // If moored or not
        bool moored_;

        // Switch to ensure that moody is only intialised once
        bool moodyStarted_;

public:

    //- Runtime type information
    TypeName("moodyR");

    // Constructors

        //- Construct from components
        moodyR
        (
               const word& name,
            const dictionary& sDoFRBMRDict
        );

        //- Construct and return a clone
        virtual autoPtr<sixDoFRigidBodyMotionRestraint> clone()
const
        {
            return autoPtr<sixDoFRigidBodyMotionRestraint>
            (
                new moodyR(*this)
            );
        }

    //- Destructor
    virtual ~moodyR();

    // Member Functions

        //- Calculate the restraint position, force and moment.
        //  Global reference frame vectors.
        void restrain
        (
            const sixDoFRigidBodyMotion& motion,
            vector& restraintPosition,
            vector& restraintForce,
            vector& restraintMoment
        );

        //- Update properties from given dictionary
       bool read(const dictionary& sDoFRBMRCoeff);

        //- Write
        void write(Ostream&) const;
};
```

```
void Foam::sixDoFRigidBodyMotionRestraints::moodyR::restrain
(
        const sixDoFRigidBodyMotion& motion,
        vector& restraintPosition,
        vector& restraintForce,
        vector& restraintMoment
)
{
        ...

        if ( moored_ )
        {
                // Compute values at present motion state
                double attachInfo[3*noOfAttachments_]; // Always 3D
        List<point> currentPositions(attachPts_);

                for ( int ii = 0; ii<noOfAttachments_; ++ii)
                {
                        // Transform the starting points to present
motion state
                        currentPositions[ii] =
motion.transform(attachPts_[ii]);

                        // Fill attachInfo with transformed values
                        attachInfo[ii*3] =   currentPositions[ii].x();
                        attachInfo[ii*3+1] = currentPositions[ii].y();
                        attachInfo[ii*3+2] = currentPositions[ii].z();
                }

                // Send info to moody
                double allForces[3*noOfAttachments_];

                moodySolve( attachInfo , allForces ,
motion.state0().time() , motion.state().time());
```

```
                // Compute moment and total force
                vector tmpVector(vector::zero);

        for ( int ii = 0; ii<noOfAttachments_; ++ii)
                {
                        tmpVector[0] = -allForces[ii*3];
                        tmpVector[1] = -allForces[ii*3+1];
                        tmpVector[2] = -allForces[ii*3+2];

                        restraintForce += tmpVector;

                        restraintMoment += (currentPositions[ii] -
motion.centreOfRotation())^tmpVector;

                        Info<< " Mooring force at point " << ii << ",
position: " << currentPositions[ii] << " is : " << tmpVector
                         << " Total force: " << restraintForce
                        << " Total moment: " << restraintMoment
                        << endl;
                }

                // End mooring loop with assigning rP as
centreOrRotation. No added moment should be computed //
                restraintPosition = motion.centreOfRotation();
        }

} // end of restrain member
```

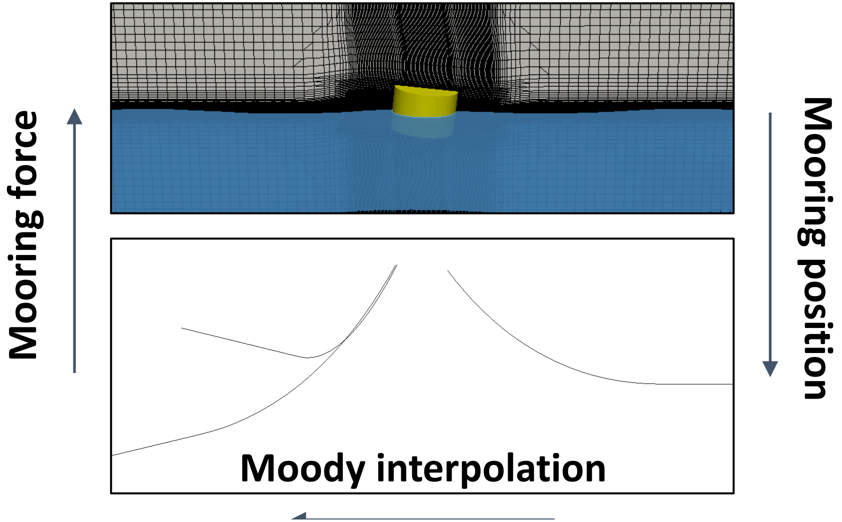
```
restraints
{
      mooring
      {
            sixDoFRigidBodyMotionRestraint      moodyR;

            attachmentPoints     (
                                       (0.6 0.6 0.3)
                                       (0.6 0.4 0.3)
                                       (0.4 0.4 0.3)
                                       (0.4 0.6 0.3)
                                 );

            mooringSystemFilename        "moodyInput.txt";

      }
}
```

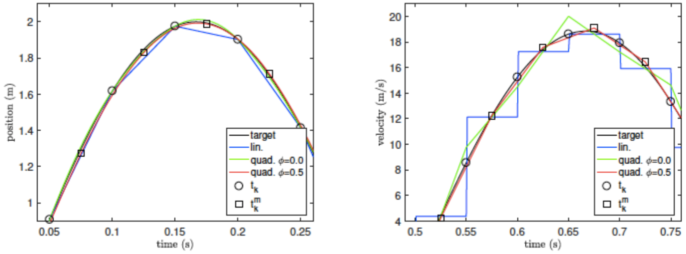
```
%------------------------------------------------------------------------%
%-------------------------- Time settings -------------------------------%
%------------------------------------------------------------------------%
startTime = 0;        %   [s]     Start time
endTime   = 1;        %   [s]     End time of simulation
timeStep = 5e-5;      %   [s]     Time step size
saveInterval = 0.1;               % save results every xx time interval
timeStepScheme='RK3';             %'RK3'
lagTimeFraction = 0.5;
courantNo = 0.9; % compute time step size from Courant no. (CFL condition)

%------------------------------------------------------------------------%
%------------------------- General settings -----------------------------%
%------------------------------------------------------------------------%
gravity = 1;             % [-]       Gravity is 1=on, 0=off
waterLevel = 0.52;       % [m]       z-coordinate of mean water level
waterDensity = 1e3;      % [kg/m??]   Density of water
airDensity = 0;          % [kg/m??]   Density of air
dimensionNumber = 3;     % [-]       1D, 2D or 3D : 1, 2 or 3

%------------------------------------------------------------------------%
%------------------------- Ground model input  --------------------------%
%------------------------------------------------------------------------%
% ground.type = 'springDampGround';
% ground.level = -50;
% ground.dampingCoeff = 1;
% ground.frictionCoeff = 1;
% ground.vc = 0.1;
% ground.stiffness = 1e6;

%------------------------------------------------------------------------%
%-------------------------- Type definition -----------------------------%
%------------------------------------------------------------------------%
cableType1.diameter = 0.01;
cableType1.gamma0 = 1;
cableType1.CDn = 1.5;
cableType1.CDt = 0.1;
cableType1.CMn = 1.0;
cableType1.materialModel.type = 'bilinearCable';
cableType1.materialModel.EA = 100; % specific input to material model.

%------------------------------------------------------------------------%
%---------------------------- Geometry ----------------------------------%
%------------------------------------------------------------------------%
vertexLocations = {
                      1    [2.5 2.5 0];
                      2    [0.0 0.0 0.0];
                      3    [2.5 -1.5 0];
                      4    [0.0 0.0 0.0];
                      5    [-1.5 -1.5 0];
                      6    [0.0 0.0 0.0];
                      7    [-1.5  2.5  0];
                      8    [0.0 0.0  0.0];
                  };
```

```
%----- Object definitions -----%
cableObject1.name = 'cable1';
cableObject1.typeNumber = 1;
cableObject1.startVertex = 1; %
cableObject1.endVertex = 2; %
cableObject1.length = 1.0; %
cableObject1.IC.type = 'Prestrain';
cableObject1.IC.esp0 = 0.1;
cableObject1.mesh.N = 5;
cableObject1.mesh.P = 4;

cableObject2.name = 'cable2';
cableObject2.typeNumber = 1;
cableObject2.startVertex = 3; %
cableObject2.endVertex = 4; %
cableObject2.length = 1.0; %
cableObject2.IC.type = 'Prestrain';
cableObject2.IC.esp0 = 0.1;
cableObject2.mesh.N = 5; %
cableObject2.mesh.P = 4;

cableObject3.name = 'cable3';
cableObject3.typeNumber = 1;
cableObject3.startVertex = 5; %
cableObject3.endVertex = 6; %
cableObject3.length = 1.0; %
cableObject3.IC.type = 'Prestrain';
cableObject3.IC.esp0 = 0.1;
cableObject3.mesh.N = 5; %
cableObject3.mesh.P = 4;

cableObject4.name = 'cable4';
cableObject4.typeNumber = 1;
cableObject4.startVertex = 7; %
cableObject4.endVertex = 8; %
cableObject4.length = 1.0; %
cableObject4.IC.type = 'Prestrain';
cableObject4.IC.esp0 = 0.1;
cableObject4.mesh.N = 5; %
cableObject4.mesh.P = 4;

%------------------------------------------------------------------------%
%----------------------- Boundary conditions ---------------------------%
%------------------------------------------------------------------------%
bc1.vertexNumber = 1;
bc1.type = 'dirichlet'; % 'dirichlet, neumann or mixed.
bc1.mode = 'fixed';

bc2.vertexNumber = 2;
bc2.type = 'dirichlet';
bc2.mode = 'externalPoint'; % laggingQuadraticInterp';


bc3.vertexNumber = 3;
bc3.type = 'dirichlet'; % 'dirichlet, neumann or mixed.
bc3.mode = 'fixed';

bc4.vertexNumber = 4;
bc4.type = 'dirichlet';
bc4.mode = 'externalPoint'; % laggingQuadraticInterp';


bc5.vertexNumber = 5;
bc5.type = 'dirichlet'; % 'dirichlet, neumann or mixed.
bc5.mode = 'fixed';

bc6.vertexNumber = 6;
bc6.type = 'dirichlet';
```

Validation

Two timescales. Boundary condition interpolation required

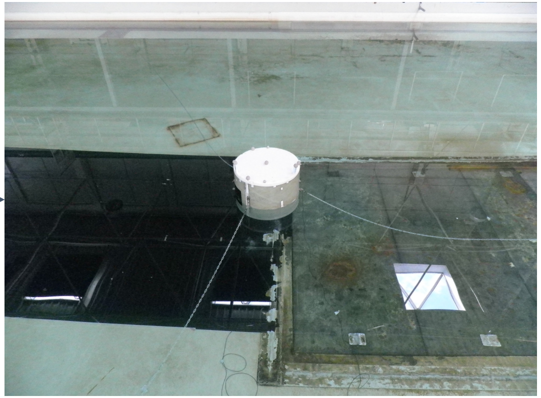


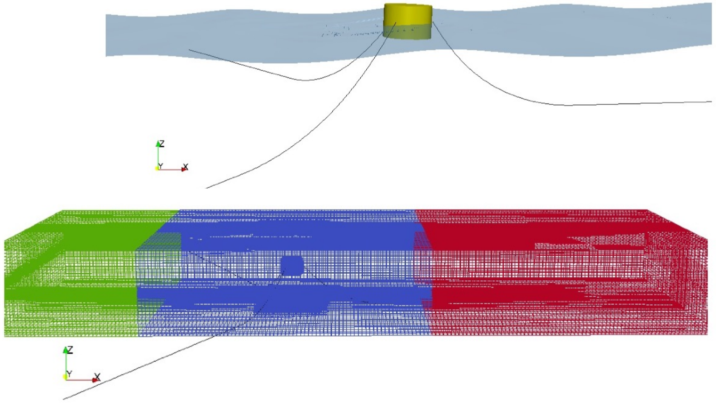


(a) Position (b) Velocity

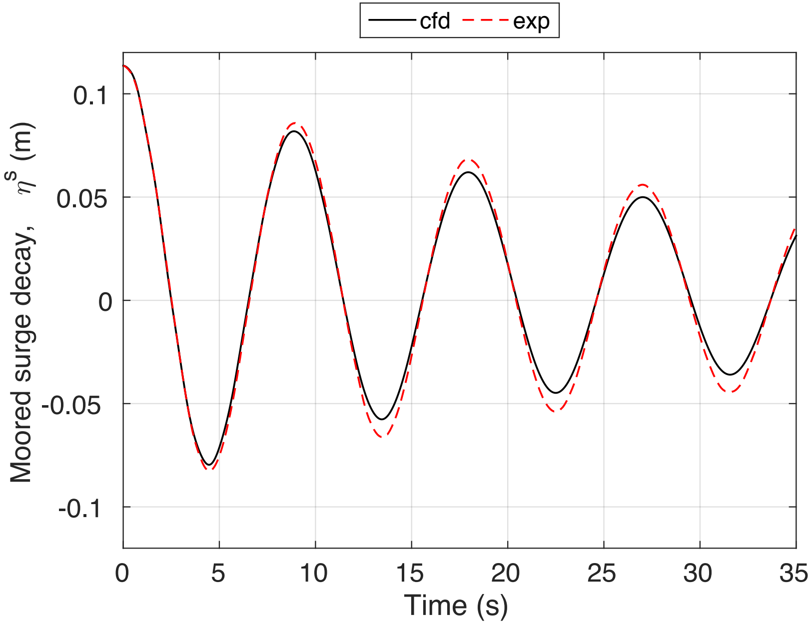
Porto test

Paredes et al 2016



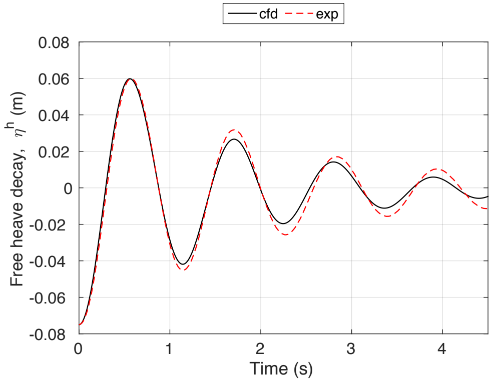


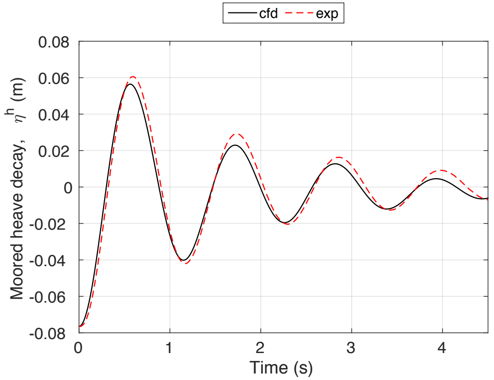
- **Surge (horizontal)**
  - *Very good agreement*
  - *All stiffness comes from the moorings*
  - *Coupled mooring validated!*



Decay tests

- **Heave (vertical)**
  - *Very good agreement both with and without moorings*
  - *Slightly more damping in cfd*



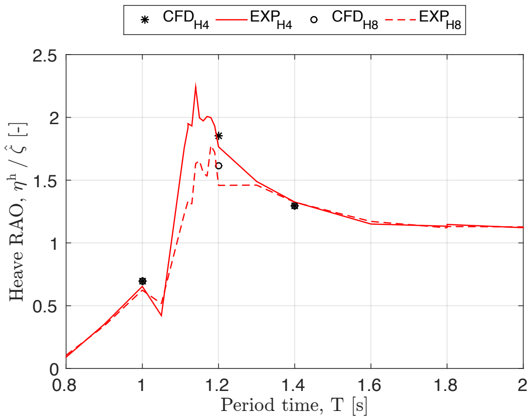


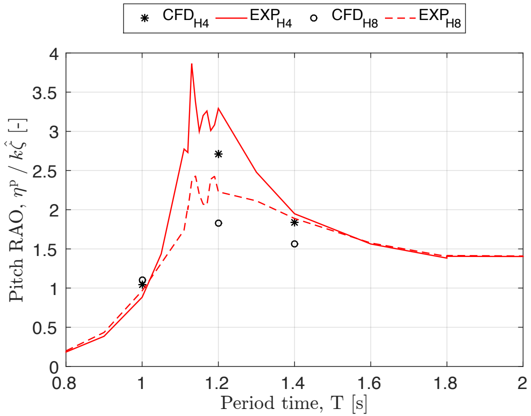
- **Motion response**
  - *Three wave periods, two wave heights*
  - *Good agreement in heave and mooring force*
  - *RAO is strongly nonlinear*
  - *Pitch is underpredicted, but influence of nonlinearity is correct*

RAO: Response Amplitude Operator

Motion amplitude / wave amplitude.

Constant for each frequency in linear theory

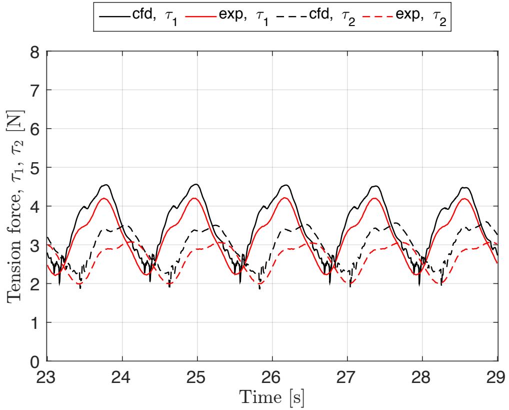




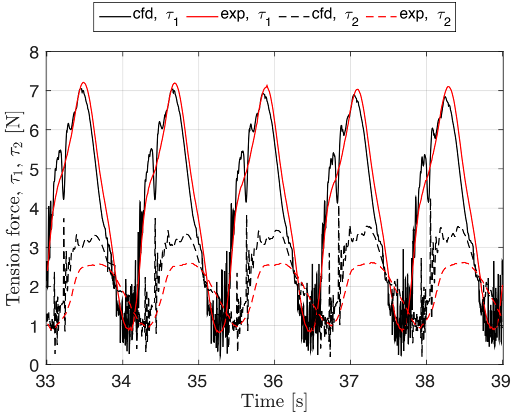
Regular waves

- **Mooring forces**
  - *Good agreement in mooring forces*
  - *Overall dynamic behaviour captured*
  - *Some discrepancies in amplitude*

T=1.2 s, *H*=4 cm



T=1.2 s, *H*=8 cm



Detailed info can be found in the following publications (open-access)

V International Conference on Computational Methods in Marine Engineering
MARINE 2013
B. Brinkmann and P. Wriggers (Eds)

SIMULATION OF MOORING CABLE DYNAMICS USING A DISCONTINUOUS GALERKIN METHOD

JOHANNES PALM*, GUILHERME MOURA PAREDES†, CLAES ESKILSSON*, FRANSISCO TAVEIRA PINTO† AND LARS BERGDAHL*

* Department of Shipping and Marine Technology
Chalmers University of Technology
SE–412 96 Gothenburg, Sweden
Email: johannes.palm@chalmers.se - Web page: www.chalmers.se

† Departamento de Engenharia Civil
Faculdade de Engenharia, Universidade do Porto
4200–465 Porto, Portugal
Email: moura.paredes@fe.up.pt - Web page: www.fe.up.pt

http://publications.lib.chalmers.se/records/fulltext/183127/local_183127.pdf

International Journal of Marine Energy 16 (2016) 83–99
Contents lists available at ScienceDirect
International Journal of Marine Energy
journal homepage: www.elsevier.com/locate/ijome

Coupled mooring analysis for floating wave energy converters using CFD: Formulation and validation

Johannes Palm [a,*], Claes Eskilsson [a], Guilherme Moura Paredes [b], Lars Bergdahl [a]
[a] Department of Shipping and Marine Technology, Chalmers University of Technology, SE-412 96 Gothenburg, Sweden
[b] Department of Civil Engineering, Faculty of Engineering, University of Porto, 4200-465 Porto, Portugal

http://www.sciencedirect.com/science/article/pii/S2214166916300327

THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Mooring Dynamics for Wave Energy Applications

A high-order discontinuous Galerkin method for cables, coupled to Reynolds averaged Navier-Stokes simulations.

JOHANNES PALM

...artment of Shipping and Marine Technology
...LMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
2017

publications.lib.chalmers.se/records/fulltext/248879/248879.pdf

Ocean Engineering 144 (2017) 266–276
Contents lists available at ScienceDirect
Ocean Engineering
journal homepage: www.elsevier.com/locate/oceaneng

An *hp*-adaptive discontinuous Galerkin method for modelling snap loads in mooring cables

Johannes Palm *, Claes Eskilsson [1], Lars Bergdahl
Department of Mechanics and Maritime Sciences, Chalmers University of Technology, SE-412 96, Gothenburg, Sweden

Keywords: Mooring cable, Snap load, Discontinuous Galerkin method, High-order, Shock-capturing

http://publications.lib.chalmers.se/records/fulltext/183127/local_183127.pdf

USERS MANUAL
version m-1.0

MOODY

JOHANNES PALM
CLAES ESKILSSON

www.sdwed.civil.aau.dk/digitalAssets/97/97522_d2.4.pdf

sixDOFRigidBodyRestraint::moodyR and libmoodyWrap.so can be downloaded from:

https:://github.com/johannep/moodyAPI