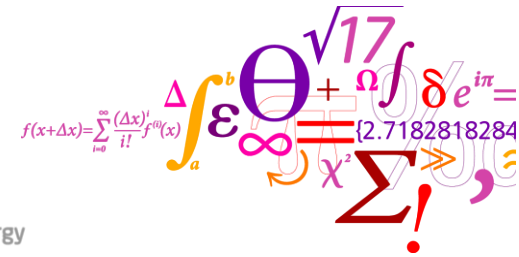
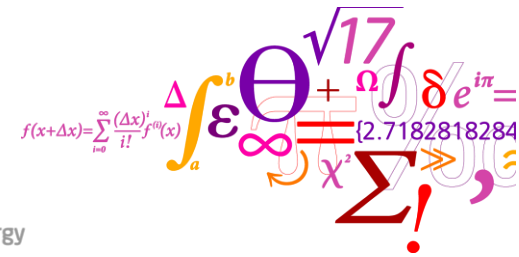
User-defined External Systems Applied to Floating structures and mooring lines

Anders M. Hansen (anmh@dtu.dk)



Risø DTU
National Laboratory for Sustainable Energy

User-defined External Systems Applied to Floating structures and mooring lines



Risø DTU
National Laboratory for Sustainable Energy

DLL Interface to external systems

Un-constrained EOMs:

$$\delta W = \sum_{i=1}^{N} \delta W_i = \sum_{i=1}^{N} \delta \vec{q}_i \cdot \vec{B}_i = 0$$

Constrained EOMs:

$$\delta W + \delta(\vec{\lambda} \cdot \vec{g}) = \delta W + \delta\vec{\lambda} \cdot \vec{g} + \delta\vec{g} \cdot \vec{\lambda} = 0 \quad for \quad \vec{g} = \vec{0}$$

Matrix EOMs:

$$\delta \vec{q}_i \cdot \left(\vec{B}_i + \nabla_{q_i}^T \vec{g} \cdot \vec{\lambda} \right) = 0 \quad for \; i = 1..N$$

$$\delta \vec{\lambda} \cdot (\vec{g}) = 0$$

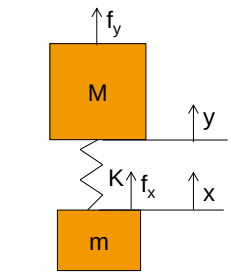
Implemented in DLL external systems !

$K =$

Coupling constraints

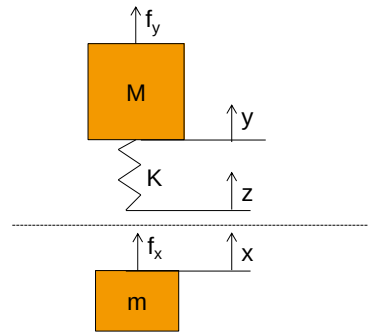
Simple example

General system



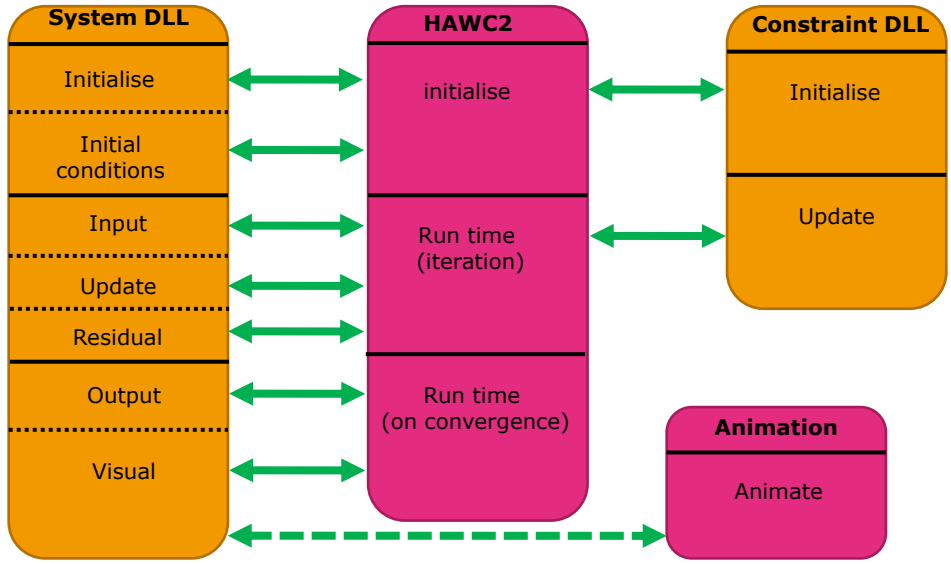
$$\begin{bmatrix} M & 0 \\ 0 & m \end{bmatrix} \begin{Bmatrix} \ddot{y} \\ \ddot{x} \end{Bmatrix} + \begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \begin{Bmatrix} y \\ x \end{Bmatrix} - \begin{Bmatrix} f_y \\ f_x \end{Bmatrix} = 0$$

HAWC2 system using constraints



$$\left[\begin{array}{cc|c|c} M & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & m & 0 \\ \hline 0 & 0 & 0 & 0 \end{array}\right] \begin{Bmatrix} \ddot{y} \\ \ddot{z} \\ \ddot{x} \\ \lambda \end{Bmatrix} + \left[\begin{array}{cc|c|c} k & -k & 0 & 0 \\ -k & k & 0 & 1 \\ \hline 0 & 0 & 0 & -1 \\ \hline 0 & 1 & -1 & 0 \end{array}\right] \begin{Bmatrix} y \\ z \\ x \\ \lambda \end{Bmatrix} - \begin{Bmatrix} f_y \\ 0 \\ f_x \\ 0 \end{Bmatrix} = 0$$

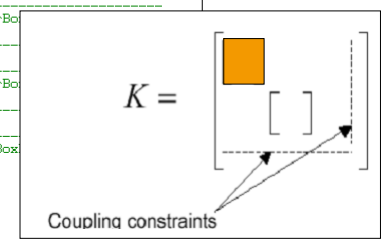
DLL Interface to external systems



DLL System (programmed by the user)

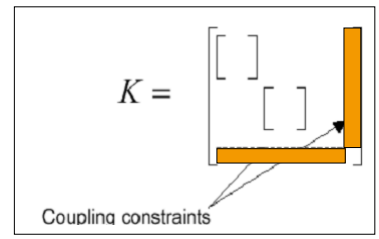
```
MODULE GearBoxDLL
!
CONTAINS
!
!+ ------------------------------------------------------------------------------
subroutine GearBoxDLL_init(pwrk,nnr, nnq, nout, nvis, nheader, sdata)
!+ ------------------------------------------------------------------------------
!DEC$ ATTRIBUTES ALIAS:"gearboxdll_init", DLLEXPORT::GearBoxDLL_init
!
!+ ------------------------------------------------------------------------------
subroutine GearBoxDLL_initcond(pwrk,x, xdot, xdot2)
!+ ------------------------------------------------------------------------------
!DEC$ ATTRIBUTES ALIAS:"gearboxdll_initcond", DLLEXPORT::GearBoxDLL_initcond
!
!+ ------------------------------------------------------------------------------
subroutine GearBoxDLL_update(pwrk, time, x, xdot, xdot2, MEFF, CEFF, KEFF)
!+ ------------------------------------------------------------------------------
!DEC$ ATTRIBUTES ALIAS:"gearboxdll_update", DLLEXPORT::GearBoxDLL_update
!
!+ ------------------------------------------------------------------------------
SUBROUTINE GearBoxDLL_visual(pwrk, flag, iodata)
!+ ------------------------------------------------------------------------------
!DEC$ ATTRIBUTES ALIAS:"gearboxdll_visual", DLLEXPORT::GearBo
!
!+ ------------------------------------------------------------------------------
SUBROUTINE GearBoxDLL_output(pwrk, time, output)
!+ ------------------------------------------------------------------------------
!DEC$ ATTRIBUTES ALIAS:"gearboxdll_output", DLLEXPORT::GearBo
!
!+ ------------------------------------------------------------------------------
SUBROUTINE GearBoxDLL_input(pwrk, time, nin, input)
!+ ------------------------------------------------------------------------------
!DEC$ ATTRIBUTES ALIAS:"gearboxdll_input", DLLEXPORT::GearBox
!
END MODULE GearBoxDLL
```

Mandatory

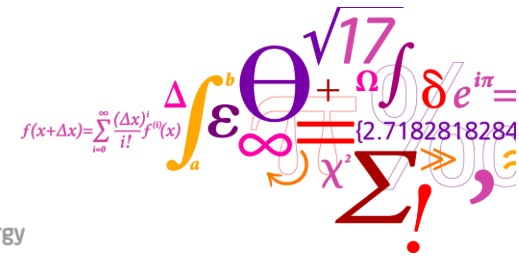


DLL Constraints (programmed by the user)

```
MODULE constraint96
CONTAINS
!
!*******************************************************************************
SUBROUTINE constraint96_init(pwrk, itask, var1, var2, var3, var4, var5, strID)
!*******************************************************************************
! Tasks: Initialisation, pointer setting and reading of input
!
!*******************************************************************************
SUBROUTINE constraint96_update(pwrk, time)
!*******************************************************************************
! Tasks: Calc. constraint vector and gradients
END MODULE
!
```



User-defined External Systems Applied to Floating structures and mooring lines



Risø DTU
National Laboratory for Sustainable Energy

Dynamic Mooring Line Modeling in Hydro-Aero-Servo-Elastic Wind Turbine Simulations

Bjarne S. Kallesøe
Anders M. Hansen

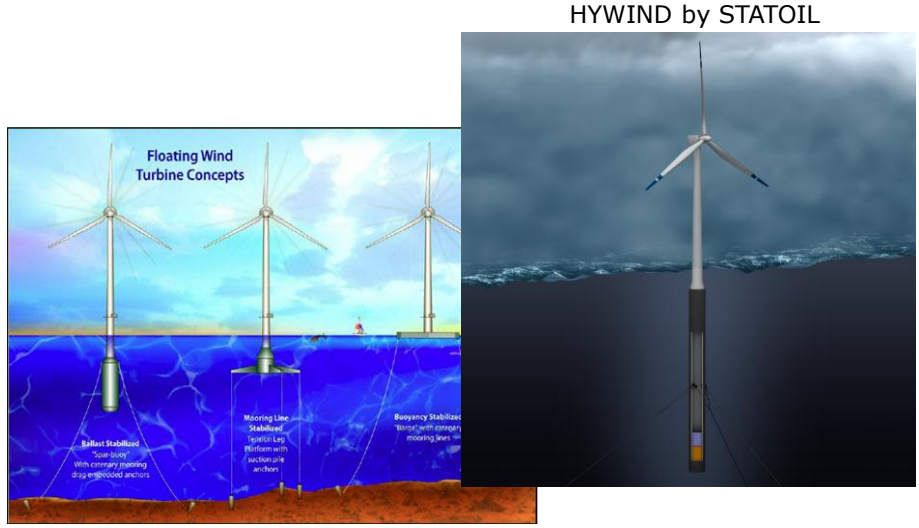
ACKNOWLEDGMENTS
The work has been funded by the Danish Energistyrelsen under contract EUDP 63011-0190.

A special acknowledgment goes to STATOIL for helpful discussion of mooring systems and providing the mooring setup and MIMOSA results.

Risø DTU
National Laboratory for Sustainable Energy

Mooring line modeling

HYWIND by STATOIL



This work

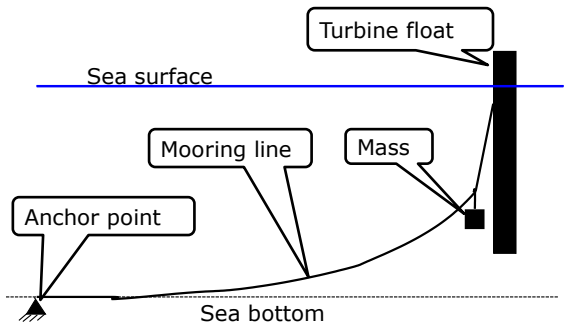
Why:

- To analyze the effect of an advanced mooring line model on the wind turbine loads.

How:

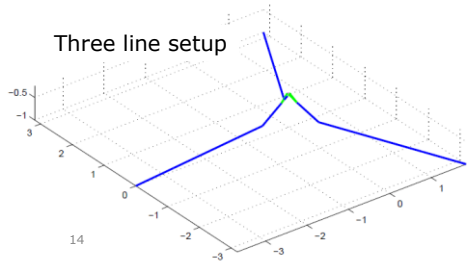
- Current mooring lines is modeled in a quasi-static framework.
- Add a dynamic mooring line model to the hydro-aero-servo-elastic code HAWC2.
- Compare turbine loads from a selected number of load cases for the different mooring line model complexities.

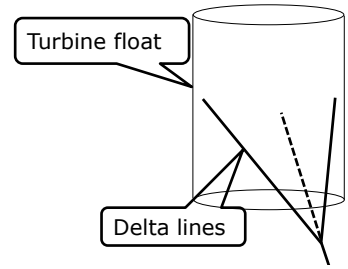
Catenary mooring system



Mooring line properties.

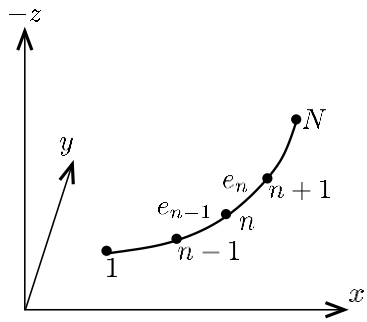
- Different section with uniform properties e.g. chain, synthetic rope, etc.
- Concentrated masses.
- Each line section modeled as one body
- Bodies connected by ball-joint constraints





Nonlinear stiffness term

One line segment with uniform properties.



Length of element:

$$L_n = \sqrt{(x_{n-1} - x_n)^2 + (y_{n-1} - y_n)^2 + (z_{n-1} - z_n)^2}$$

Green strain:

$$\epsilon_G = \frac{L_n^2 - L_{n,0}^2}{2L_{n,0}^2}$$

Axial force in element:

$$f = EA\epsilon_G$$

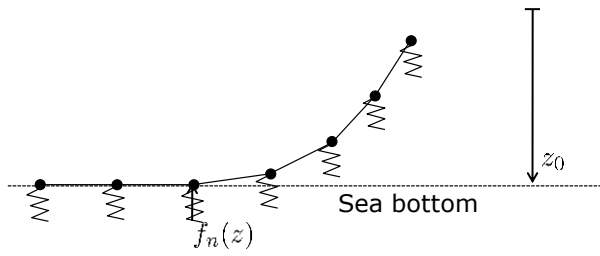
Element stiffness matrix:

$$K_e = f/L_n \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

Nodal elastic forces:

$$K_e \begin{bmatrix} \mathbf{x}_n \\ \mathbf{x}_{n+1} \end{bmatrix} = \begin{bmatrix} -f\delta \\ f\delta \end{bmatrix}, \quad \delta = \begin{bmatrix} \frac{x_{n+1}-x_n}{L_n} \\ \frac{y_{n+1}-y_n}{L_n} \\ \frac{z_{n+1}-z_n}{L_n} \end{bmatrix}$$

Bottom contact



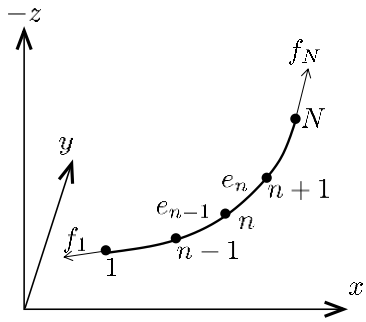
$$f_n(z) = \begin{cases} 0 & \text{if } z < z_0 \\ K\left((z - z_0)^2 + (z - z_0)\right) & \text{if } z \geq z_0 \end{cases}$$

Equations of motion

Unconstrained equation of motion:

$$\mathbf{M}\ddot{\mathbf{x}}(t) + \mathbf{K}(\mathbf{x},t)\mathbf{x}(t) - F_{gravity} - F_{buoyancy} - F_{drag}(\mathbf{x},\dot{\mathbf{x}},t) - F_{bottom} = \text{residual}$$

One line segment with uniform properties.



Constraint forces from constraint conditions:

1) distance from first node on first line segment to mooring point = 0.

2) distance from node N of one line segment to node 1 of the next segment = 0.

3) distance from node N of last line segment to node n on a HAWC2 body = 0.

Implementation in *.htc file

External body

```
begin   ext_sys        ;  2
   module  elasticbar        ;
   name    line_1_1          ;
   dll .\Release\ESYSMooring.DLL  ;
   ndata  10    ;
   data    20    ;   NELEM
   data    0     ;   L
   data    1.0  0.9   ;  ma, mw
   data    -80.6  0.0  100.0 ; x1(1:3)
   data    -61.3  0.0   80.0  ; x2(1:3)
   data    10.8  4.2 ; 0.5*rho*D*[Cd, Cd_axial]
   data    6.0E+09  ;   EA
   data    0  0  ;
   data    100 0.01 0.5  ; z0, Kbottom, Dbottom
   data    0.02  ; time step
end ext_sys         ;
```

External constraint 1

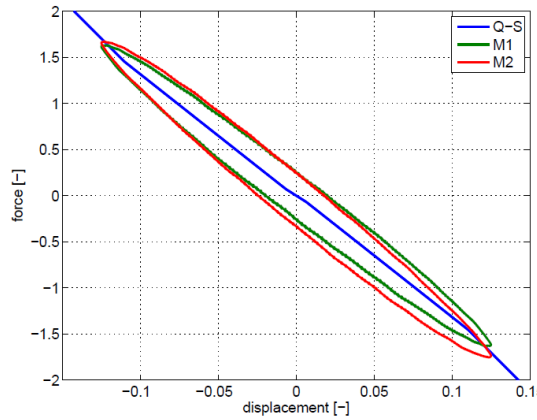
```
begin dll;
    ID 2 ;
    dll .\Release\ESYSMooring.DLL ;
    init CSTRBarsFixedToBody_init;
    update CSTRBarsFixedToBody_update;
    neq 3;
    nbodies 1;
    nesys 1;
    mbdy_node floater 9;
    esys_node line_1_7 8 ;
  end dll;
```

External constraint 2

```
begin dll;
    ID ;
    dll .\Release\ESYSMooring.DLL ;
    init   cstrbarfixedtobar_init;
    update cstrbarfixedtobar_update;
    neq 3;
    nbodies 0;
    nesys 2;
    esys_node line_1_1 21 ;
    esys_node line_1_2 1  ;
  end dll;
```

Effect of dynamic mooring line model

Horizontal force for horizontal oscillations of the float (50 s period).



The mooring restoring force is normalized by the rated aerodynamic thrust for the particular turbine and the displacement by the rotor diameter.

Eigenvalue analysis including External Systems

Full set (non-linear)

$$\begin{aligned} \mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} + \nabla\mathbf{g}^T\lambda - \mathbf{F} &= \mathbf{0} \\ \mathbf{g} &= \mathbf{0} \end{aligned}$$

Linearisation

Full set (linear)

$$\begin{aligned} \mathbf{M}_l\,\delta\ddot{\mathbf{q}} + \mathbf{C}_l\,\delta\dot{\mathbf{q}} + \mathbf{K}_l\,\delta\mathbf{q} + \nabla\mathbf{g}^T\,\delta\lambda - \delta\mathbf{F} &= \mathbf{0} \\ \nabla\mathbf{g}\,\delta\mathbf{q} &= \mathbf{0} \end{aligned}$$

Future Systems

- The plan is to make external systems publically available from the HAWC2 web page (keep an eye on www.hawc2.dk) – for some systems, including source code.

- Future systems might include:
  - Mooring models of various complexities
    - Dynamic
    - Quasi-static
    - Table-lookup
  - Drive-train models of various complexities
    - 9 DOF model of planetary gear stage.
    - Generel gear wheel tool box including gear teeth interaction
  - Earthquake modelling (prescribed acceleration of a body origin)
  - Large floating (flexible) foundations based on WAMIT-like hydrodynamics
  - Super element import (exported as generalised models from, e.g. FEM)