Modifications to HydroDyn to support Jacket platforms and the new FAST modularization framework

# Project Goals

The purpose of this work is to extend the existing NWTC Hydrodynamics simulation module (HydroDyn) to include:

* modeling of jacket platforms
* refactoring the existing codebase to fit into the new FAST modularization framework
* removal of the mooring system from the HydroDyn module
* refactoring the module so that all platform types can be modeled in a more uniform way: fixed monopile, tripod, TLP, floating spars, jacket platforms, gravity bases, etc.
* This version only implements the loosely-coupled version of HydroDyn
* Generate a new user’s guide
* Generate a new theory manual
* Create a stand-alone driver program and associated unit tests which exercise the full functionality of the module

# Module Architecture

Driver Layer

HydroDyn

Waves

WAMIT

Current

Morison

Conv\_Radiation

HydroDyn\_Input+

+ Does not follow template

HydroDyn\_Output

SS\_Radiation

**Implemented Template Subroutines (Loose Coupling)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Module** | **\*\_UpdateStates** | **\*\_CalcOutput** | **\*\_CalcConstrStateResidual** | **\*\_CalcContStateDeriv** | **\*\_UpdateDiscState** |
| HydroDyn |  |  |  |  |  |
| Current |  |  |  |  |  |
| Waves |  |  |  |  |  |
| WAMIT |  |  |  |  |  |
| Conv\_Radiation |  |  |  |  |  |
| SS\_Radiation |  |  |  |  |  |
| Morison |  |  |  |  |  |

All modules implement \*\_Init, \*\_End, \*\_Pack, and \*\_Unpack.

For loose coupling, none of the \*\_Jacobian\* subroutines are implemented.

## FAST / Driver Module

This module is responsible for:

* Calling into the HydroDyn User-Input Module in order to parse hydrodynamics-related user-inputs from a file.
* Calls the HydroDyn module’s initialization routine
* Calling the time-step related HydroDyn module subroutines
* Calling the HydroDyn module clean-up subroutines.
* Calling the HydroDyn module’s Pack and Unpack subroutines.
* Passing the HydroDyn module the necessary output, input, state, and parameter arguments for each of the above subroutines.
* ~~Calls subroutines in the HydroDyn File output routines in order to write requested outputs to a file.~~

Data which is passed to HydroDyn

Gravity

Water density

Platform DOF flags

## HydroDyn Module

This module is the principal interface between the hydrodynamics simulation and the calling program (FAST or a driver program). This module follows the FAST modularization template, Template.f90 and Template\_Types.f90. The module serves the following roles:

* Responsible for initializing the parameters, p, and states (x, xd, zGuess), managed by the HydroDyn sub-modules as well as calling initialization code managed by sub-modules
* Implements time-step-related subroutines as outlined in the template:
  + Updates states
  + Passes time-step outputs which includes forces, moments, and added-mass matrices which are generated by the WAMIT and Morison and Conv\_Radiation and SS\_Radiation sub-modules
* Responsible for the deallocation of all module data and initiates the deallocation of sub-module data
* Implements all the template subroutines except for the Jacobian routines.

## Waves Sub-module

This module encapsulates the wave kinematics physics. This module follows the FAST modularization template, Template.f90 and Template\_Types.f90.

## Current Sub-module

## WAMIT Sub-module

This module implements the potential flow theory hydrodynamics. Currently, the module will incorporate WAMIT-generated data corresponding to the diffraction, hydrostatics, and radiation. This module follows the FAST modularization template, Template.f90 and Template\_Types.f90.

## Conv\_Radiation Sub-module

## SS\_Radiation Sub-module

## Morison Elements Sub-module

This module implements Morison element hydrodynamics. This module follows the FAST modularization template, Template.f90 and Template\_Types.f90.

## HydroDyn Input Sub-module

This module encapsulates the reading of user-settings associated with the hydrodynamics module. This module does not follow the FAST modularization template.

## HydroDyn Ouput Sub-module

This module encapsulates the writing of user-requested outputs associated with the hydrodynamics module. This module does not follow the

# Platform Modeling

Include Figure 2 from Jason’s paper showing platform reference point and global coordinate system.

Include a Figure of a fixed-bottom jacket platform and show exploded view of a joint / super member +

# Governing Equations

## Nomenclature

All velocities, accelerations,forces, moments, and added-mass matrices are expressed in terms of the global coordinate system.

### Temporary Variables

*  member or element index
*  node index
*  Length of the ith slave cylinder of a super member
*  Point buoyancy forces at the end cap of the ith member which connects to the super member
*  Outer radius of the super member’s master cylinder
*  Outer radius of the ith slave cylinder which is part of a super member
*  Marine-growth thickness on the master cylinder
*  Marine-growth thickness on ith slave cylinder of a super member
*  Volume of ith cylinder which is a part of the super member
*  Relative velocity, , 3x1 vector

### Parameters

*  Density of the flooding fluid, constant specified via user-input or defaults to water density
*  Density of marine growth, constant specified via user-input
*  Water density, constant specified via user-input
*  Linear acceleration of the fluid, time and (X,Y,Z)-dependent, 3x1 vector
*  Area of super member, constant calculated during initialization of module
*  Potential flow theory-generated added mass 6x6 matrix due to radiation problem
*  Added-mass coefficient, constant specified via user-input
*  Viscous-drag coefficient, constant specified via user-input
*  Incident-wave excitation force applied to platform reference point, time-dependent
*  Gravity, constant
*  Radiation kernel from potential flow theory, user-supplied constant 6x6 matrix
*  Length of master cylinder of a super member, constant determined at initialization
*  Outer radius of structural member, constant specified via user-input
*  Thickness of structural member, constant specified via user-input
*  Thickness of structural member at element end, constant specified via user-input
*  Thickness of structural member at element start, constant specified via user-input
*  Marine-growth thickness, constant specified via user-input
*  Volume of member (super member), constant calculated during initialization of module
*  Interior cavity volume of member (super member), constant calculated during initialization of module
*  added volume due to marine growth, constant calculated during initialization of module
*  Linear velocity of the fluid, time and (X,Y,Z)-dependent, 3x1 vector

### Inputs

The following inputs are specified at each node on the hydrodynamics mesh.

*  Rotational structural velocities, 3x1 vector, currently not used.
*  Marker direction cosine 3x3 matrix, transforming local coordinates to global coordinates
*  Translational structural velocities, 3x1 vector
*  Position relative to platform-reference-point (0,0,0)

### Outputs

*  Added-mass, 6x6 matrix
*  Added-mass due to flooding, 6x6 matrix
*  Added-mass due to Morison’s equations, 6x6 matrix
*  Added-mass due to marine growth, 6x6 matrix
*  Added-mass due to radiation problem applied to the platform reference point, 6x6 matrix
*  Total forces and moments (not including added mass) applied to node j, 6x1 vector
*  Buoyancy forces and moments applied to node, 6x1 vector
*  Drag forces and moments applied to node, 6x1 vector
*  Dynamic pressure forces and moments applied to node, 6x1 vector
*  Buoyancy forces and moments due to flooding applied to node, 6x1 vector
*  Hydrostatic forces applied to the platform reference node, 6x1 vector
*  Inertial forces and moments applied to node, 6x1 vector
*  Marine growth-related forces and moments applied to node, 6x1 vector
*  Radiation memory-effect force applied to platform reference point, 6x1 vector
*  Incident-wave excitation force applied to platform reference point, 6x1 vector

### Discrete States

*  Radiation damping history, dependent on time and structure velocities.

## Global coordinate system:

* The global axes are represented by the unit vectors , , and .
* The origin is set at the mean sea level, the center of the structure, with axis positive upward.
* The positive axis is along the nominal (zero-degree) wave propagation direction.

## Structural member local coordinate system:

* Axes are represented by the unit vectors , , and .
* The origin is set at the center of the structural member.
* The local axis is along the cylinder axis, directed from the start point to the end point. The start point is defined as the end point that has a lower coordinate value. If the two end points have the same coordinate value, then the one that has the lower coordinate value is the start point. If the two end points have the same and coordinate value, then the one that has the lower coordinate value is the start point.
* The local axis is parallel to the global plane, positive along the nominal wave propagation direction. If the cylinder’s axis is along the global direction, then the local axis is parallel to the plane, and positive along the negative global direction.
* The local axis follows right hand rule.

## Local to Global transformation

For regular members, the cylinder expression in global coordinate system can be found as follows:



where  and  are the start and end joints of the member in global coordinate system of the member, and  is the direction cosine matrix of the member axis and can be obtained as follows:



where  and . When  and , the  matrix can be found as follows:

if  then



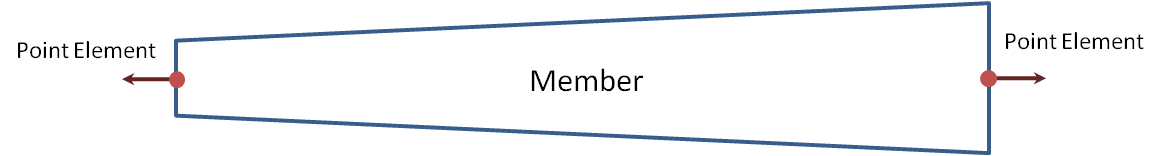
else



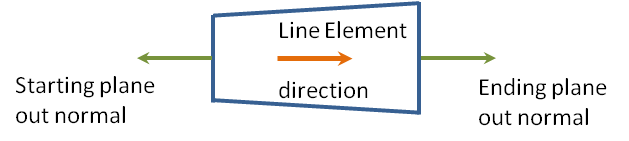
## Super member local coordinate system:

* Axes are represented by the unit vectors , , and .
* 
* 

### End Point Element



### Line Element



# Loads Calculations

## Forces and added mass at platform reference point (potential flow theory)

 (1)

 (2)

 (3)

 (4)

 (5)

 (6)

 (7)

 (8)

The state for time step n is simply the structural velocities for the previous  time steps. Or,

 (9)

Therefore, at time step n, we can compute the n+1 state, such that,

 (10)

The radiation step size, , and the radiation kernel, *K* , are simply module parameters.

## For distributed forces on structural members which are not modeled using potential flow theory

 (11)

All of these forces except the buoyancy forces, , and , are applied directly to a mesh node. The buoyancy forces are computed for each elements, and then the node-based force is found by weighting and summing all element forces connected to the mesh node.

 (12)

All of these weights are applied directly to a mesh node.

 (13)

 (14)

 (15)

 (16)

 (18)

What is C in this equation?

, (19)

The force  has the magnitude of the weight of the marine growth per unit length, but is in the opposite direction as the weight force. We include this force due to …

(20)



 (21)

 (22)

 (23)

NOTE: with current definition of direction cosine matrix , at initialization, but if we later let the structural code update C and recomputed the hydrodynamic forces, then we need to differentiate the two.

## For distributed forces on structural members which are modeled using potential flow theory

Note: Marine growth is not allowed for members modeled using potential flow theory

 (24)

 (25)

 (26)

 (27)

 (28)

## For Lumped forces on the ends of structural members which are not modeled using potential flow theory

 (29)

 (30)

 where is the outward facing normal in global coordinates. (31)

Equation 31 is the total lumped drag at a joint, since the simulation mesh consists of M co-located lumped nodes,  is actually , for each simulation node.

 (32)

 (33)

 We are ignoring marine growth on ends of members. (34)

 (35)

 (36)

 (37)

 (38)

 (39)

equation 39 is for the upper left quadrant of the 6x6 added mass matrix, the rest of the entries are zero

Equation 39 must be computed using all elements connected to a common joint (no super member). So we need to know which elements share a common joint node.

[0] (40)

## For Lumped forces on the ends of structural members which are modeled using potential flow theory

 (41)

 (42)

 (43)

Equation 43 is the total lumped drag at a joint, since the simulation mesh consists of M co-located lumped nodes,  is actually , for each simulation node.

 (44)

 (45)

## For Lumped forces associated with super members which are not modeled using potential flow theory

 (46)

 (47)

 (48)

 (49)

 (50)

 ?? (51)

 ?? (52)

 (53)

 (54)

 (55)

## For Lumped forces associated with super members which are modeled using potential flow theory

 (56)

 (57)

?? (58)

 (59)

 (60)

# HydroDyn Outputs (For output to file, not for use by FAST)

Outputs are setup/initialized and generated by the HydroDyn module. The HydroDyn module in turn makes calls into the HydroDyn\_Output module to perform the actual work.

Output to the file is generated by the driver program using the HydroDyn\_Output module.

%NMOutputs

%MOutLst

TYPE, PUBLIC :: Morison\_MOutput

INTEGER :: MemberID ! member ID in the master list

INTEGER :: NOutLoc ! number of output locations along the member (normalized)

REAL(ReKi), ALLOCATABLE :: NodeLocs(:)

INTEGER :: MemberIDIndx = -1

END TYPE Morison\_MOutput

NJOutputs

JOutLst

JOutLstIndx

InitInp%OutRootName

InitInp%OutList(:) : Names of all requested outputs

P%UnOutFile

P%OutParam

P%Delim

P%NumOuts

P%OutSFmt

Y%WriteOutput(0:MaxOutPts)

## HDOut\_ChkOutLst( OutList, y, p, …)

Checks that the requested channel name is a valid channel name in the master list

This sets:

P%OutParm data

P%OutParam(I)%Name = OutList(I)

## HDOut\_MapOutputs

This subroutine assigns the current timestep’s data to the appropriate index in the master y%WriteOutput array

## Output handling Workflow

Begin

Read Outputs section of the HydroDyn Input File

Initialize the file outputs data structures

Attach the output quantities at time, t, to the master output array

Write the requested file outputs for time, t.

# HydroDyn Mesh

* Mesh data includes:
  + Geometry (a list of nodes), connectivity (a list of elements), and node properties (position, orientation, etc.)
  + Mesh data will utilize the MeshType structure of the ModMesh.f90 module and the associated types of MeshTypes.f90
  + The Hydrodynamics will be modeled using both point elements and line elements.
  + If the floating portion of the platform is being modeled using linear potential-flow theory, there will be a single point element (and node) associated with this portion of the model. This node will be located at the platform-reference-point (0,0,0), and will have an Identity orientation matrix.
  + For straight, cylindrical, structural members there will be:
    - N+1 nodes associated with N line elements of equal length. The loads at these nodes will be formulated as distributed.
    - 2 nodes associated with 2 point elements, one for each end of the structural member, will have lumped loads.
  + If structural members meet (connect) to one another, there may be a single point element (and node) which represents the intersecting volume where these members meet. There will be lumped loads at this node.
  + There will be one mesh containing all point elements and a second mesh containing all line elements.
  + The module will generate three sibling meshes from a single source mesh.
  + The source mesh will be used to represent module outputs and will contain a collection of scalar fields which are in common with all sibling meshes. These fields (generated for each node in the mesh) are:
    - A switch field indicating whether the mesh node represents:
      * The platform reference point where linear potential-flow loads are to be applied
      * Platform geometry which is modeled using potential-flow theory
      * Platform geometry which is not modeled using potential-flow theory
    - Radius of structural member, 
    - Thickness of structural member, 
    - Added-mass coefficient, 
    - Viscous-drag coefficient, 
    - Marine-growth thickness, 
    - Density of marine growth, 
    - Density of the flooding fluid, 
    - Position relative to platform-reference-point (0,0,0), 
    - Orientation relative to inertial coordinate system, 
    - Translational and rotational structural velocities, 
    - Forces and Moments applied to node, 
    - Added-mass associated with node (which is in addition to and independent of any added-mass generated by the structural-dynamics module), 
    - Velocity of the fluid, 
    - Acceleration of the fluid, 
  + Sibling meshes will be generated to model:
    - Inputs
    - States
    - Parameters
  + Sibling meshes will contain scalar fields unique to the data they represent (inputs, states, or parameters)

The platform geometry will be specified using the NWTC ModMesh module. This module contains the necessary data structures and subroutines to represent, query, and manipulate the hydrodynamic-related platform geometry, parameters, and states.

Initializing the mesh will be performed by an internal HydroDyn initialization subroutine. See the Module Initialization section.

The final mesh will be generated using the following calls:

CALL MeshCreate(hydroMesh,

Since the module’s state variables are represented by a structure, the mesh sibling associated with the states will be attached via the DiscStates field to the HydroDyn\_stateType structure

TYPE, PUBLIC :: HydroDyn\_StateType

TYPE(HydroDyn\_ContinuousStateType) :: ContStates ! The continuous states

TYPE(HydroDyn\_DiscreteStateType) :: DiscStates ! The discrete states

TYPE(HydroDyn\_ConstraintStateType) :: ConstrStates ! The constraint states

! Define any data that are not considered actual "states" here:

! e.g. data used only for optimization purposes (indices for searching in an array, copies of previous calculations

! of output at a given time, etc.)

! If you have loose coupling with a variable-step integrator, store the actual time associated with the continuous

! states here (eg., REAL(DbKi) :: ContTime):

END TYPE HydroDyn\_StateType

TYPE, PUBLIC :: HydroDyn\_DiscreteStateType

! Define discrete (nondifferentiable) states here:

TYPE(MeshType) :: DiscStateMesh !mesh holding discrete state information

END TYPE HydroDyn\_DiscreteStateType

# Future Work

* Super Member implementation
* Support for reporting the added mass force as an output instead of the added mass matrix
* Wave stretching models