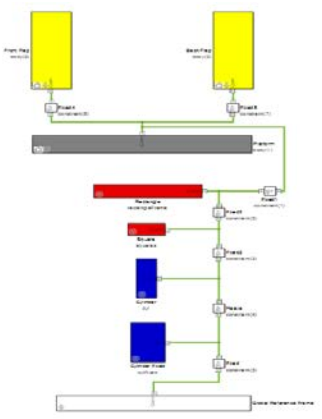
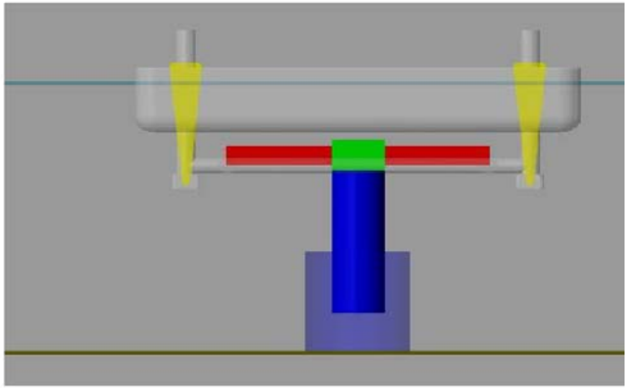
U.S. DEPARTMENT OF ENERGY | Energy Efficiency & Renewable Energy

FOSWEC wave tank testing and WEC-Sim simulation





# WEC-Sim Training Course

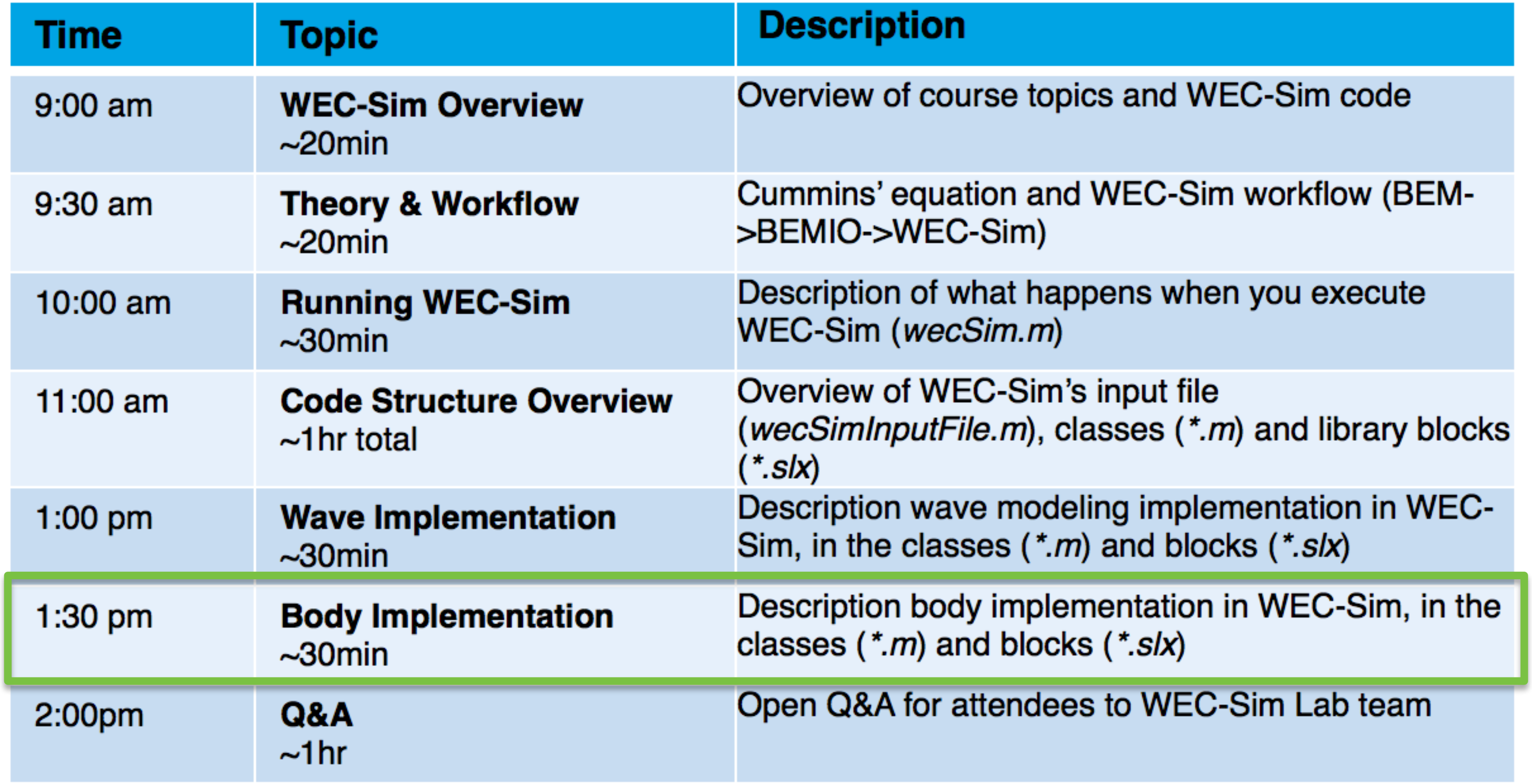
## for users and developers

**August 17, 2017**

Yi-Hsiang Yu (NREL)

Kelley Ruehl (Sandia)

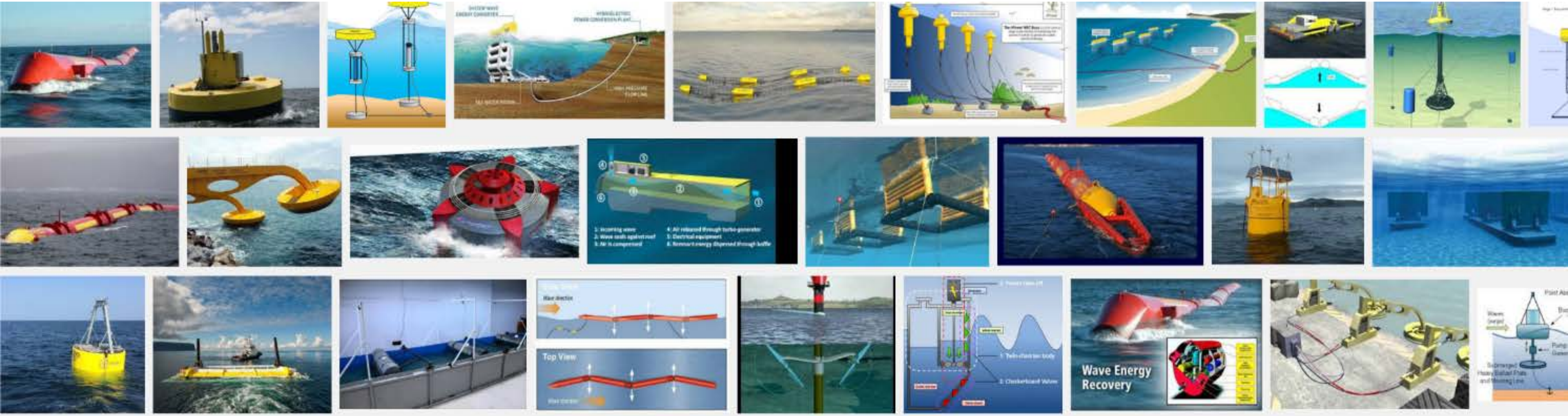
# Course Agenda



| Time | Topic | Description |
|---|---|---|
| 9:00 am | **WEC-Sim Overview**<br>~20min | Overview of course topics and WEC-Sim code |
| 9:30 am | **Theory & Workflow**<br>~20min | Cummins' equation and WEC-Sim workflow (BEM->BEMIO->WEC-Sim) |
| 10:00 am | **Running WEC-Sim**<br>~30min | Description of what happens when you execute WEC-Sim (*wecSim.m*) |
| 11:00 am | **Code Structure Overview**<br>~1hr total | Overview of WEC-Sim's input file (*wecSimInputFile.m*), classes (**.m*) and library blocks (**.slx*) |
| 1:00 pm | **Wave Implementation**<br>~30min | Description wave modeling implementation in WEC-Sim, in the classes (**.m*) and blocks (**.slx*) |
| 1:30 pm | **Body Implementation**<br>~30min | Description body implementation in WEC-Sim, in the classes (**.m*) and blocks (**.slx*) |
| 2:00pm | **Q&A**<br>~1hr | Open Q&A for attendees to WEC-Sim Lab team |

# WEC-Sim Webinar

U.S. DEPARTMENT OF ENERGY | Energy Efficiency & Renewable Energy

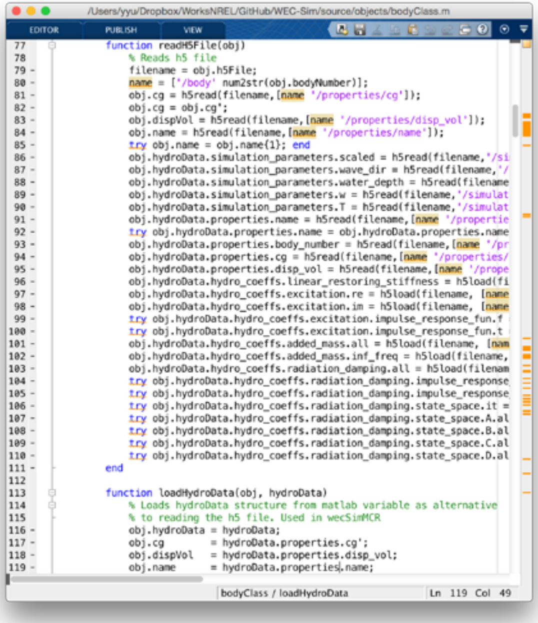


# Body Implementation

Yi-Hsiang (NREL)

# Body Class

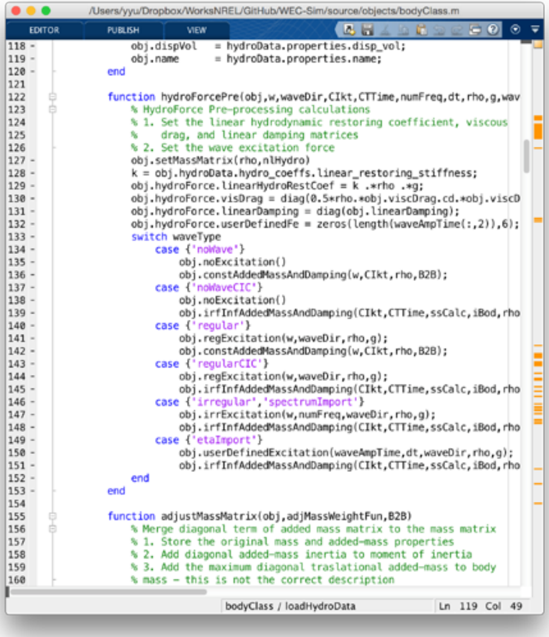
## Load Hydrodynamic Data


/Users/yyu/Dropbox/WorksNREL/GitHub/WEC-Sim/source/objects/bodyClass.m
EDITOR
PUBLISH
VIEW

```
    function readH5File(obj)
        % Reads h5 file
        filename = obj.h5File;
        name = ['/body' num2str(obj.bodyNumber)];
        obj.cg = h5read(filename,[name '/properties/cg']);
        obj.cg = obj.cg';
        obj.dispVol = h5read(filename,[name '/properties/disp_vol']);
        obj.name = h5read(filename,[name '/properties/name']);
        try obj.name = obj.name{1}; end
        obj.hydroData.simulation_parameters.scaled = h5read(filename,'/si
        obj.hydroData.simulation_parameters.wave_dir = h5read(filename,'/
        obj.hydroData.simulation_parameters.water_depth = h5read(filename
        obj.hydroData.simulation_parameters.w = h5read(filename,'/simulat
        obj.hydroData.simulation_parameters.T = h5read(filename,'/simulat
        obj.hydroData.properties.name = h5read(filename,[name '/propertie
        try obj.hydroData.properties.name = obj.hydroData.properties.name
        obj.hydroData.properties.body_number = h5read(filename,[name '/pr
        obj.hydroData.properties.cg = h5read(filename,[name '/properties/
        obj.hydroData.properties.disp_vol = h5read(filename,[name '/prope
        obj.hydroData.hydro_coeffs.linear_restoring_stiffness = h5load(fi
        obj.hydroData.hydro_coeffs.excitation.re = h5load(filename, [name
        obj.hydroData.hydro_coeffs.excitation.im = h5load(filename, [name
        try obj.hydroData.hydro_coeffs.excitation.impulse_response_fun.f
        try obj.hydroData.hydro_coeffs.excitation.impulse_response_fun.t
        obj.hydroData.hydro_coeffs.added_mass.all = h5load(filename, [nam
        obj.hydroData.hydro_coeffs.added_mass.inf_freq = h5load(filename,
        obj.hydroData.hydro_coeffs.radiation_damping.all = h5load(filenam
        try obj.hydroData.hydro_coeffs.radiation_damping.impulse_response
        try obj.hydroData.hydro_coeffs.radiation_damping.impulse_response
        try obj.hydroData.hydro_coeffs.radiation_damping.state_space.it =
        try obj.hydroData.hydro_coeffs.radiation_damping.state_space.A.al
        try obj.hydroData.hydro_coeffs.radiation_damping.state_space.B.al
        try obj.hydroData.hydro_coeffs.radiation_damping.state_space.C.al
        try obj.hydroData.hydro_coeffs.radiation_damping.state_space.D.al
    end

    function loadHydroData(obj, hydroData)
        % Loads hydroData structure from matlab variable as alternative
        % to reading the h5 file. Used in wecSimMCR
        obj.hydroData = hydroData;
        obj.cg        = hydroData.properties.cg';
        obj.dispVol   = hydroData.properties.disp_vol;
        obj.name      = hydroData.properties.name;
```

bodyClass / loadHydroData    Ln 119 Col 49

## Hydrodynamic Force Calculations


/Users/yyu/Dropbox/WorksNREL/GitHub/WEC-Sim/source/objects/bodyClass.m
EDITOR
PUBLISH
VIEW

```
        obj.dispVol   = hydroData.properties.disp_vol;
        obj.name      = hydroData.properties.name;
    end

    function hydroForcePre(obj,w,waveDir,CIkt,CTTime,numFreq,dt,rho,g,wav
        % HydroForce Pre-processing calculations
        % 1. Set the linear hydrodynamic restoring coefficient, viscous
        %    drag, and linear damping matrices
        % 2. Set the wave excitation force
        obj.setMassMatrix(rho,nlHydro)
        k = obj.hydroData.hydro_coeffs.linear_restoring_stiffness;
        obj.hydroForce.linearHydroRestCoef = k .*rho .*g;
        obj.hydroForce.visDrag = diag(0.5*rho.*obj.viscDrag.cd.*obj.viscD
        obj.hydroForce.linearDamping = diag(obj.linearDamping);
        obj.hydroForce.userDefinedFe = zeros(length(waveAmpTime(:,2)),6);
        switch waveType
            case {'noWave'}
                obj.noExcitation()
                obj.constAddedMassAndDamping(w,CIkt,rho,B2B);
            case {'noWaveCIC'}
                obj.noExcitation()
                obj.irfInfAddedMassAndDamping(CIkt,CTTime,ssCalc,iBod,rho
            case {'regular'}
                obj.regExcitation(w,waveDir,rho,g);
                obj.constAddedMassAndDamping(w,CIkt,rho,B2B);
            case {'regularCIC'}
                obj.regExcitation(w,waveDir,rho,g);
                obj.irfInfAddedMassAndDamping(CIkt,CTTime,ssCalc,iBod,rho
            case {'irregular','spectrumImport'}
                obj.irrExcitation(w,numFreq,waveDir,rho,g);
                obj.irfInfAddedMassAndDamping(CIkt,CTTime,ssCalc,iBod,rho
            case {'etaImport'}
                obj.userDefinedExcitation(waveAmpTime,dt,waveDir,rho,g);
                obj.irfInfAddedMassAndDamping(CIkt,CTTime,ssCalc,iBod,rho
        end
    end

    function adjustMassMatrix(obj,adjMassWeightFun,B2B)
        % Merge diagonal term of added mass matrix to the mass matrix
        % 1. Store the original mass and added-mass properties
        % 2. Add diagonal added-mass inertia to moment of inertia
        % 3. Add the maximum diagonal traslational added-mass to body
        % mass - this is not the correct description
```

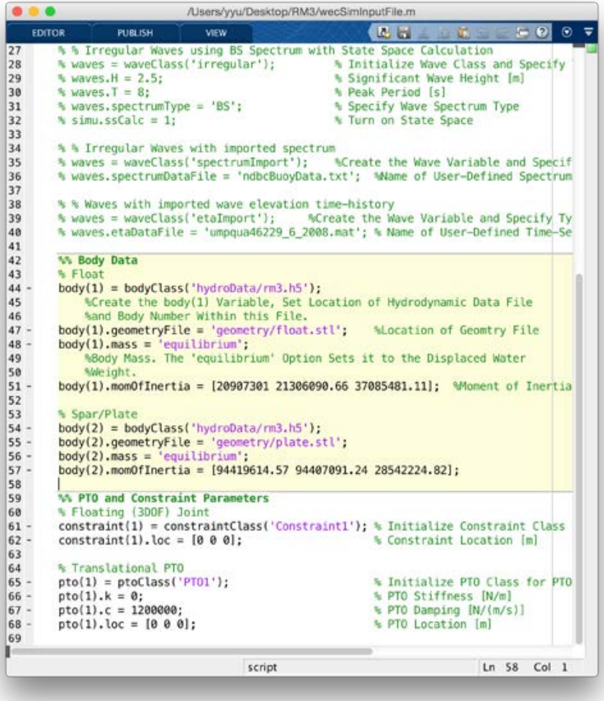
bodyClass / loadHydroData    Ln 119 Col 49

## Others

- Functions for adjusting, restoring and storing add-mass matrix during and after the simulation
- Functions for ParaView Visualization

# Body Implementation

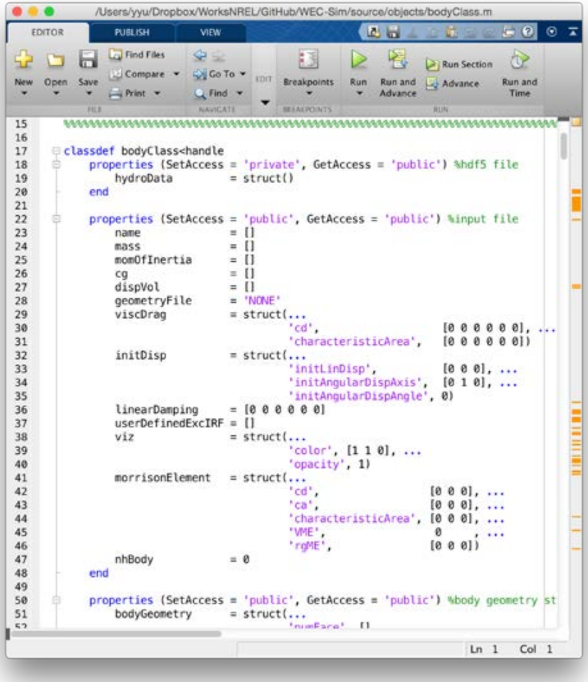
## WEC-Sim input file (wecSimInputFile.m)



```
% % Irregular Waves using BS Spectrum with State Space Calculation
% waves = waveClass('irregular');            % Initialize Wave Class and Specify
% waves.H = 2.5;                             % Significant Wave Height [m]
% waves.T = 8;                               % Peak Period [s]
% waves.spectrumType = 'BS';                 % Specify Wave Spectrum Type
% simu.ssCalc = 1;                           % Turn on State Space

% % Irregular Waves with imported spectrum
% waves = waveClass('spectrumImport');       %Create the Wave Variable and Specif
% waves.spectrumDataFile = 'ndbcBuoyData.txt';  %Name of User-Defined Spectrum

% % Waves with imported wave elevation time-history
% waves = waveClass('etaImport');       %Create the Wave Variable and Specify Ty
% waves.etaDataFile = 'umpqua46229_6_2008.mat'; % Name of User-Defined Time-Se

%% Body Data
% Float
body(1) = bodyClass('hydroData/rm3.h5');
    %Create the body(1) Variable, Set Location of Hydrodynamic Data File
    %and Body Number Within this File.
body(1).geometryFile = 'geometry/float.stl';    %Location of Geomtry File
body(1).mass = 'equilibrium';
    %Body Mass. The 'equilibrium' Option Sets it to the Displaced Water
    %Weight.
body(1).momOfInertia = [20907301 21306090.66 37085481.11];  %Moment of Inertia

% Spar/Plate
body(2) = bodyClass('hydroData/rm3.h5');
body(2).geometryFile = 'geometry/plate.stl';
body(2).mass = 'equilibrium';
body(2).momOfInertia = [94419614.57 94407091.24 28542224.82];

%% PTO and Constraint Parameters
% Floating (3DOF) Joint
constraint(1) = constraintClass('Constraint1'); % Initialize Constraint Class
constraint(1).loc = [0 0 0];                    % Constraint Location [m]

% Translational PTO
pto(1) = ptoClass('PTO1');                      % Initialize PTO Class for PTO
pto(1).k = 0;                                   % PTO Stiffness [N/m]
pto(1).c = 1200000;                             % PTO Damping [N/(m/s)]
pto(1).loc = [0 0 0];                           % PTO Location [m]
```

script | Ln 58 Col 1

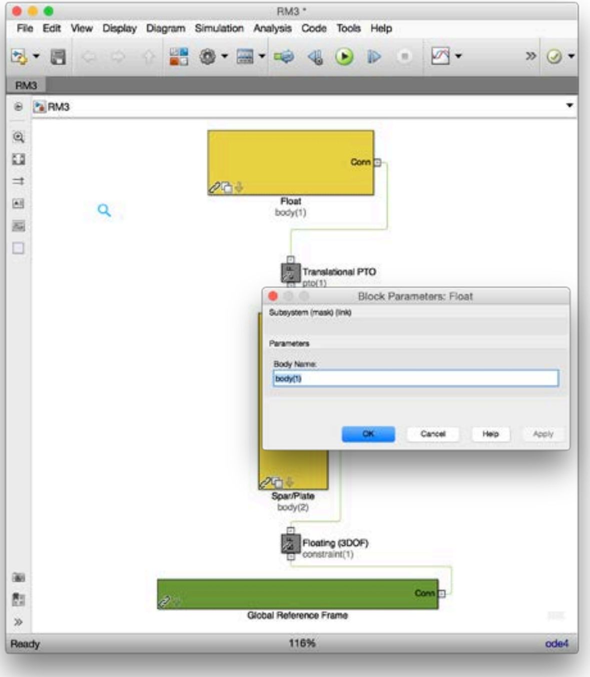
## Body Object (bodyClass.m)



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

classdef bodyClass<handle
    properties (SetAccess = 'private', GetAccess = 'public') %hdf5 file
        hydroData          = struct()
    end

    properties (SetAccess = 'public', GetAccess = 'public') %input file
        name               = []
        mass               = []
        momOfInertia       = []
        cg                 = []
        dispVol            = []
        geometryFile       = 'NONE'
        viscDrag           = struct(...
                                    'cd',                  [0 0 0 0 0 0], ...
                                    'characteristicArea',  [0 0 0 0 0 0])
        initDisp           = struct(...
                                    'initLinDisp',         [0 0 0], ...
                                    'initAngularDispAxis', [0 1 0], ...
                                    'initAngularDispAngle', 0)
        linearDamping      = [0 0 0 0 0 0]
        userDefinedExcIRF  = []
        viz                = struct(...
                                    'color', [1 1 0], ...
                                    'opacity', 1)
        morrisonElement    = struct(...
                                    'cd',                 [0 0 0], ...
                                    'ca',                 [0 0 0], ...
                                    'characteristicArea', [0 0 0], ...
                                    'VME',                 0     , ...
                                    'rgME',               [0 0 0])
        nhBody             = 0
    end

    properties (SetAccess = 'public', GetAccess = 'public') %body geometry st
        bodyGeometry       = struct(...
                                    'numFace', []
```

Ln 1 Col 1

## Body Block



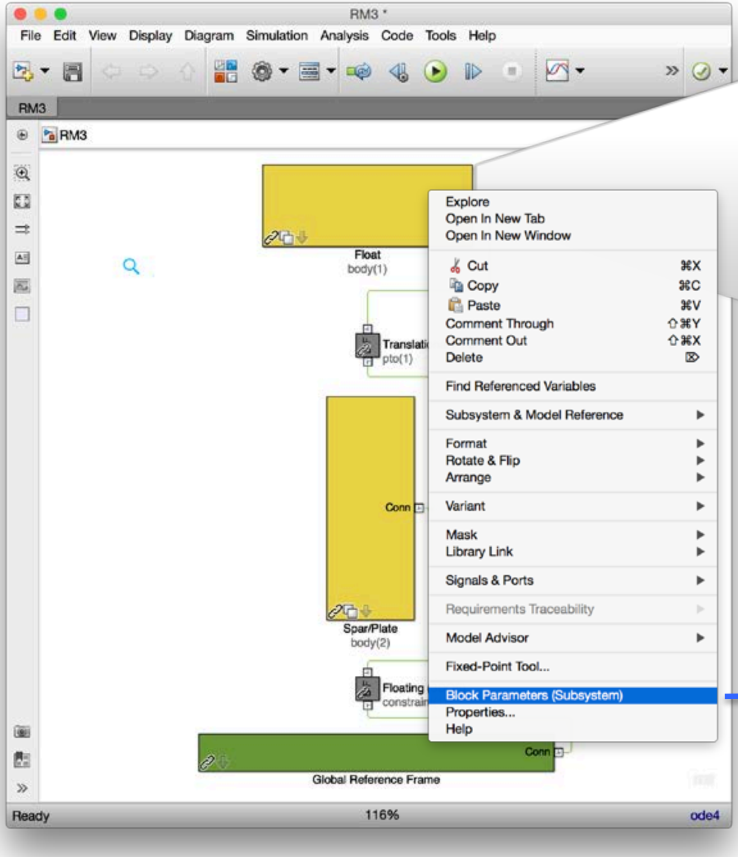
Block Parameters: Float

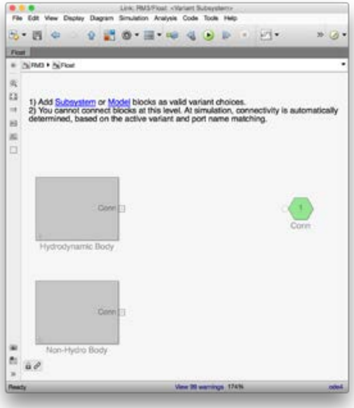
Subsystem (mask) (link)

Parameters

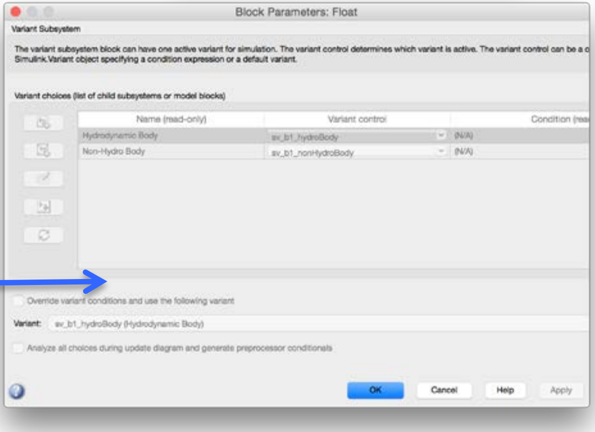
Body Name: body(1)

# Variant Subsystems




Link: RM3/Float: <Variant Subsystem>
File Edit View Display Diagram Simulation Analysis Code Tools Help
Float
RM3 ▸ Float
Conn
Hydrodynamic Body
Conn
Non-Hydro Body
1
Conn
Ready
View 39 warnings 174%
ode4

1) Add Subsystem or Model blocks as valid variant choices.
2) You cannot connect blocks at this level. At simulation, connectivity is automatically determined, based on the active variant and port name matching.



Variant Subsystem

The variant subsystem block can have one active variant for simulation. The variant control determines which variant is active. The variant control can be a c Simulink.Variant object specifying a condition expression or a default variant.

Variant choices (list of child subsystems or model blocks)

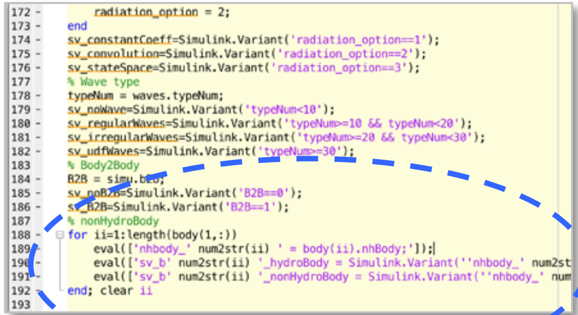
| Name (read-only) | Variant control | Condition (rea |
|---|---|---|
| Hydrodynamic Body | sv_b1_hydroBody | (N/A) |
| Non-Hydro Body | sv_b1_nonHydroBody | (N/A) |

□ Override variant conditions and use the following variant

Variant: sv_b1_hydroBody (Hydrodynamic Body)

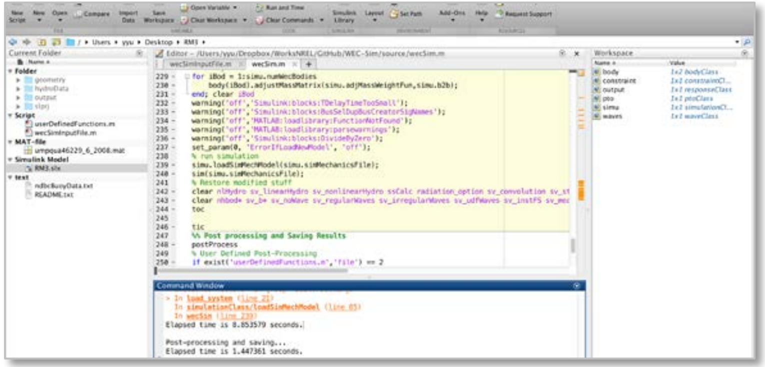
□ Analyze all choices during update diagram and generate preprocessor conditionals

wecSim .m



```
        radiation_option = 2;
    end
    sv_constantCoeff=Simulink.Variant('radiation_option==1');
    sv_convolution=Simulink.Variant('radiation_option==2');
    sv_stateSpace=Simulink.Variant('radiation_option==3');
    % Wave type
    typeNum = waves.typeNum;
    sv_noWave=Simulink.Variant('typeNum<10');
    sv_regularWaves=Simulink.Variant('typeNum>=10 && typeNum<20');
    sv_irregularWaves=Simulink.Variant('typeNum>=20 && typeNum<30');
    sv_udfWaves=Simulink.Variant('typeNum>=30');
    % Body2Body
    B2B = simu.b2b;
    sv_noB2B=Simulink.Variant('B2B==0');
    sv_B2B=Simulink.Variant('B2B==1');
    % nonHydroBody
    for ii=1:length(body(1,:))
        eval(['nhbody_' num2str(ii) ' = body(ii).nhBody;']);
        eval(['sv_b' num2str(ii) '_hydroBody = Simulink.Variant(''nhbody_' num2st
        eval(['sv_b' num2str(ii) '_nonHydroBody = Simulink.Variant(''nhbody_' num
    end; clear ii
```

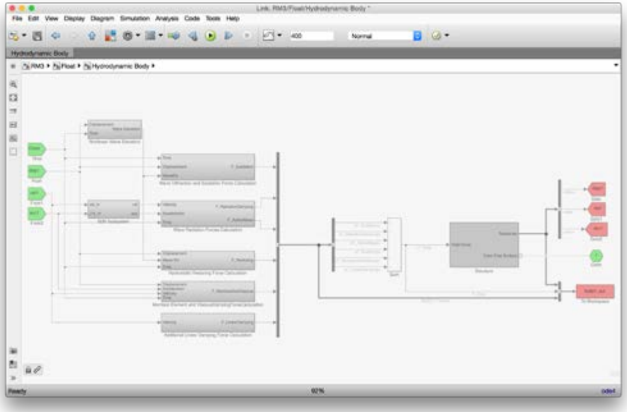
# Variant Subsystems

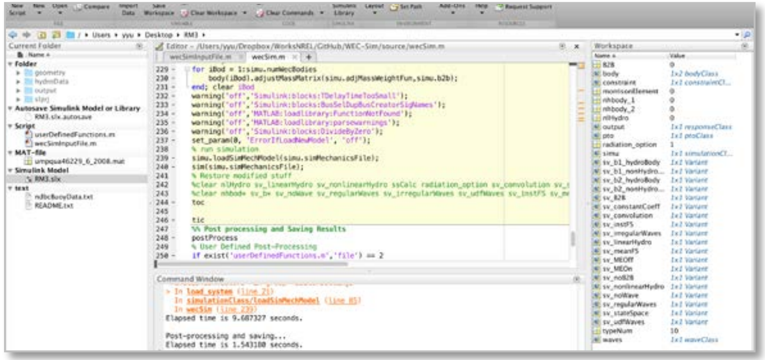


```
for iBod = 1:simu.numWecBodies
    body(iBod).adjustMassMatrix(simu.adjMassWeightFun,simu.b2b);
end; clear iBod
warning('off','Simulink:blocks:TDelayTimeTooSmall');
warning('off','Simulink:blocks:BusSelDupBusCreatorSigNames');
warning('off','MATLAB:loadlibrary:FunctionNotFound');
warning('off','MATLAB:loadlibrary:parsewarnings');
warning('off','Simulink:blocks:DivideByZero');
set_param(0, 'ErrorIfLoadNewModel', 'off');
% run simulation
simu.loadSimMechModel(simu.simMechanicsFile);
sim(simu.simMechanicsFile);
% Restore modified stuff
clear nlHydro sv_linearHydro sv_nonlinearHydro ssCalc radiation_option sv_convolution sv_st
clear nhbod* sv_b* sv_noWave sv_regularWaves sv_irregularWaves sv_udfWaves sv_instFS sv_mec
toc

tic
%% Post processing and Saving Results
postProcess
% User Defined Post-Processing
if exist('userDefinedFunctions.m','file') == 2
```

```
> In load_system (line 21)
  In simulationClass/loadSimMechModel (line 85)
  In wecSim (line 239)
Elapsed time is 8.853579 seconds.

Post-processing and saving...
Elapsed time is 1.447361 seconds.
```

| Name | Value |
|---|---|
| body | 1x2 bodyClass |
| constraint | 1x2 constraintCl... |
| output | 1x1 responseClass |
| pto | 1x1 ptoClass |
| simu | 1x1 simulationCl... |
| waves | 1x1 waveClass |

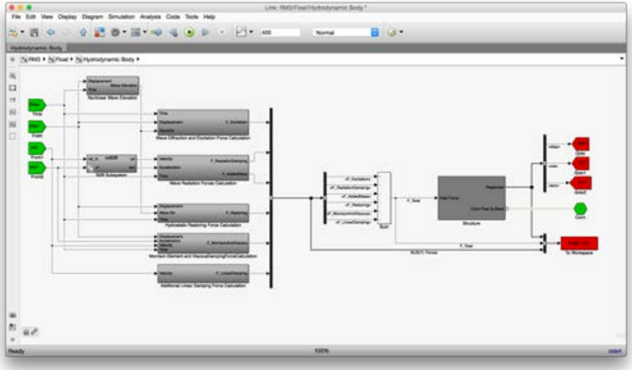




```
for iBod = 1:simu.numWecBodies
    body(iBod).adjustMassMatrix(simu.adjMassWeightFun,simu.b2b);
end; clear iBod
warning('off','Simulink:blocks:TDelayTimeTooSmall');
warning('off','Simulink:blocks:BusSelDupBusCreatorSigNames');
warning('off','MATLAB:loadlibrary:FunctionNotFound');
warning('off','MATLAB:loadlibrary:parsewarnings');
warning('off','Simulink:blocks:DivideByZero');
set_param(0, 'ErrorIfLoadNewModel', 'off');
% run simulation
simu.loadSimMechModel(simu.simMechanicsFile);
sim(simu.simMechanicsFile);
% Restore modified stuff
%clear nlHydro sv_linearHydro sv_nonlinearHydro ssCalc radiation_option sv_convolution sv_s
%clear nhbod* sv_b* sv_noWave sv_regularWaves sv_irregularWaves sv_udfWaves sv_instFS sv_me
toc

tic
%% Post processing and Saving Results
postProcess
% User Defined Post-Processing
if exist('userDefinedFunctions.m','file') == 2
```

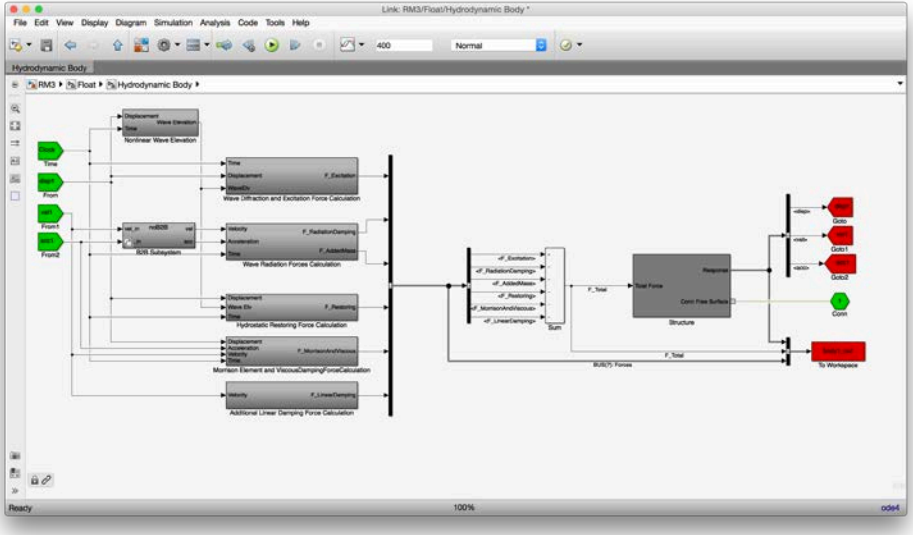
```
> In load_system (line 21)
  In simulationClass/loadSimMechModel (line 85)
  In wecSim (line 239)
Elapsed time is 9.687327 seconds.

Post-processing and saving...
Elapsed time is 1.543180 seconds.
```

| Name | Value |
|---|---|
| B2B | 0 |
| body | 1x2 bodyClass |
| constraint | 1x2 constraintCl... |
| morisonElement | 0 |
| nhbody_1 | 0 |
| nhbody_2 | 0 |
| nlHydro | 0 |
| output | 1x1 responseClass |
| pto | 1x1 ptoClass |
| radiation_option | 1 |
| simu | 1x1 simulationCl... |
| sv_b1_hydroBody | 1x1 Variant |
| sv_b1_nonHydro... | 1x1 Variant |
| sv_b2_hydroBody | 1x1 Variant |
| sv_b2_nonHydro... | 1x1 Variant |
| sv_B2B | 1x1 Variant |
| sv_constantCoeff | 1x1 Variant |
| sv_convolution | 1x1 Variant |
| sv_instFS | 1x1 Variant |
| sv_irregularWaves | 1x1 Variant |
| sv_linearHydro | 1x1 Variant |
| sv_meanFS | 1x1 Variant |
| sv_MEOff | 1x1 Variant |
| sv_MEOn | 1x1 Variant |
| sv_noB2B | 1x1 Variant |
| sv_nonlinearHydro | 1x1 Variant |
| sv_noWave | 1x1 Variant |
| sv_regularWaves | 1x1 Variant |
| sv_stateSpace | 1x1 Variant |
| sv_udfWaves | 1x1 Variant |
| typeNum | 10 |
| waves | 1x1 waveClass |



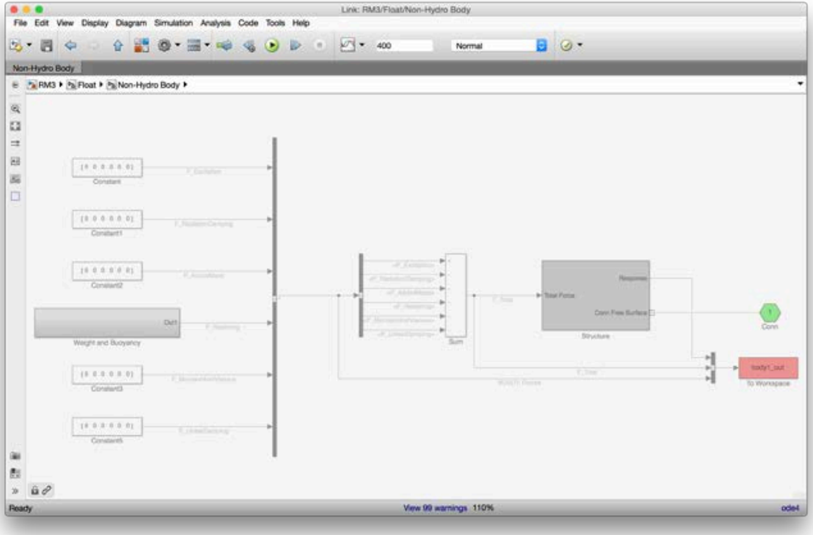
# Body Block

## Hydrodynamic Body Block



- Include blocks for calculating all the different forcing terms

## Non-Hydro Block

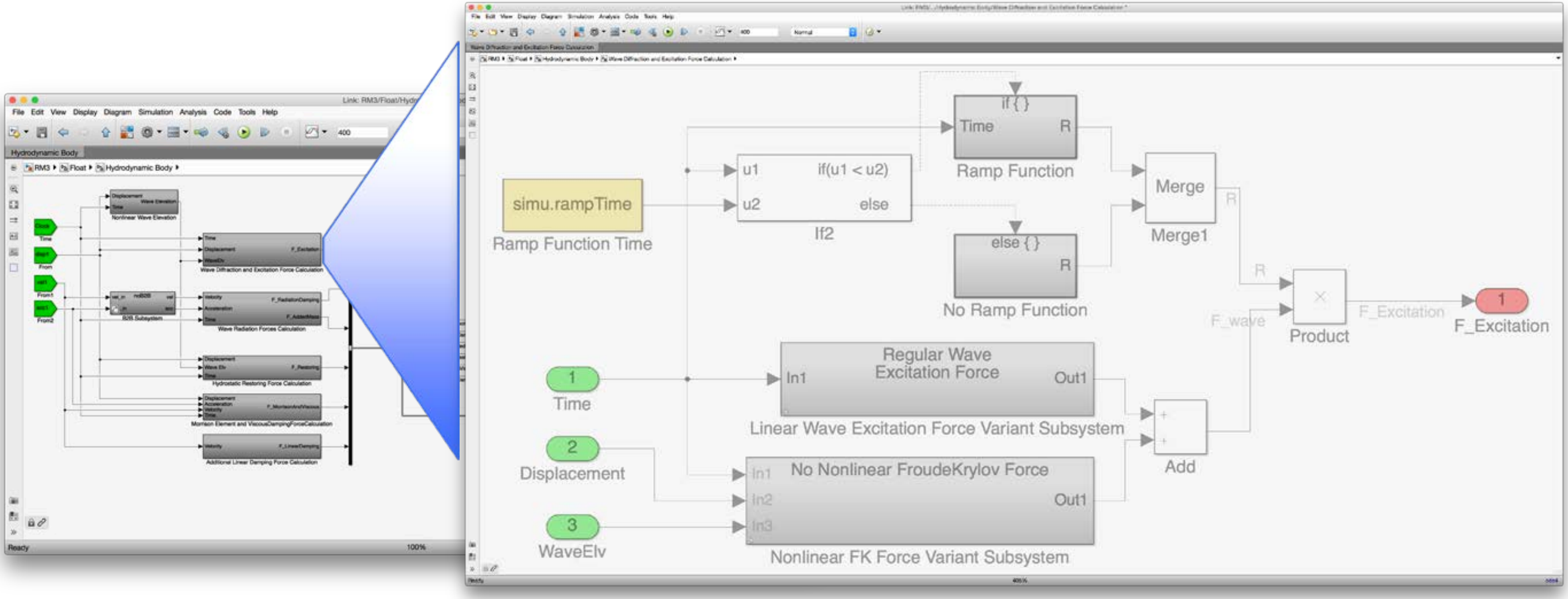


- Everything is zero except for weight and buoyancy

Note: Connection forces between multiple bodies from the joint/PTO are handled by Simscape Multibody

# Hydrodynamic Body Block:

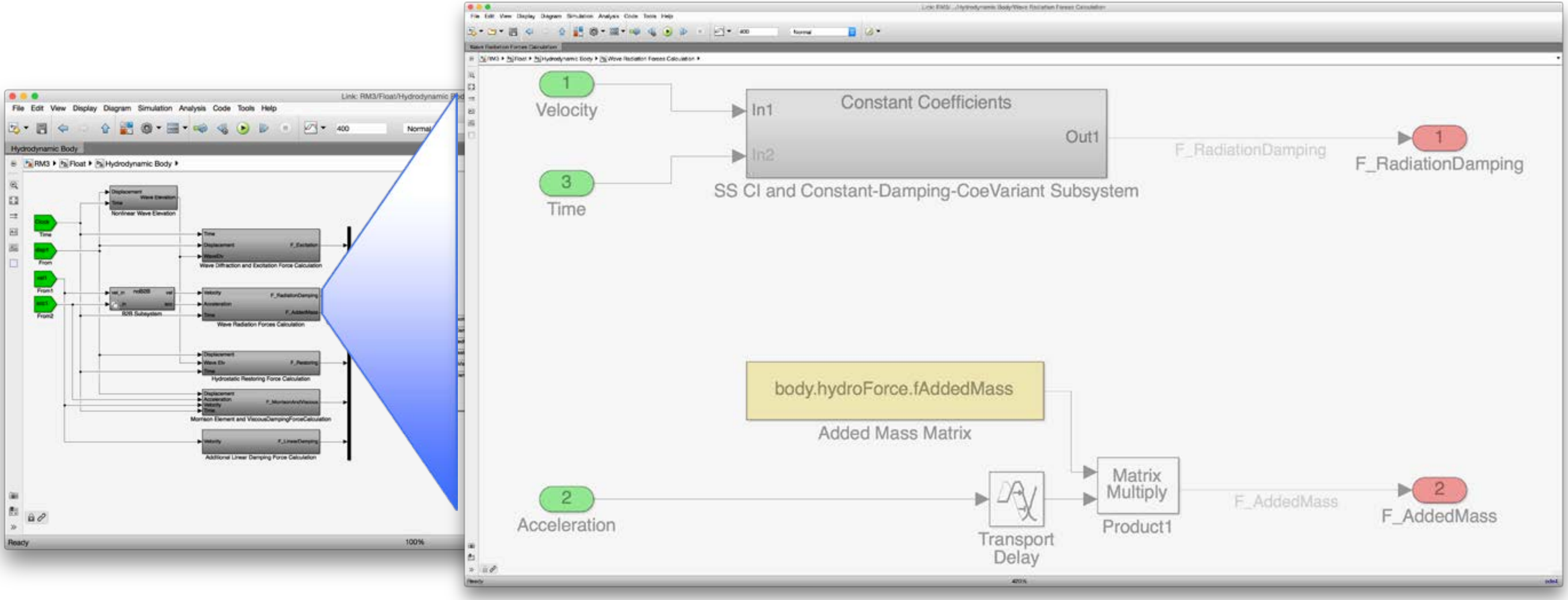
## Wave Diffraction and Excitation Force Calculation Block


File Edit View Display Diagram Simulation Analysis Code Tools Help
Hydrodynamic Body
RM3 ▸ Float ▸ Hydrodynamic Body ▸
Nonlinear Wave Elevation
Wave Diffraction and Excitation Force Calculation
Wave Radiation Forces Calculation
B2B Subsystem
Hydrostatic Restoring Force Calculation
Morison Element and ViscousDampingForceCalculation
Additional Linear Damping Force Calculation
Ready
100%
Wave Diffraction and Excitation Force Calculation
RM3 ▸ Float ▸ Hydrodynamic Body ▸ Wave Diffraction and Excitation Force Calculation ▸
simu.rampTime
Ramp Function Time
u1
u2
if(u1 < u2)
else
If2
if { }
Time
R
Ramp Function
else { }
R
No Ramp Function
Merge
Merge1
R
Product
F_wave
F_Excitation
1
F_Excitation
1
Time
2
Displacement
3
WaveElv
Regular Wave Excitation Force
In1
Out1
Linear Wave Excitation Force Variant Subsystem
No Nonlinear FroudeKrylov Force
In1
In2
In3
Out1
Nonlinear FK Force Variant Subsystem
Add

- Pre-calculated waves.waveAmpTime is not used for wave diffraction and excitation force calculation, except when user-defined time-series option is used.

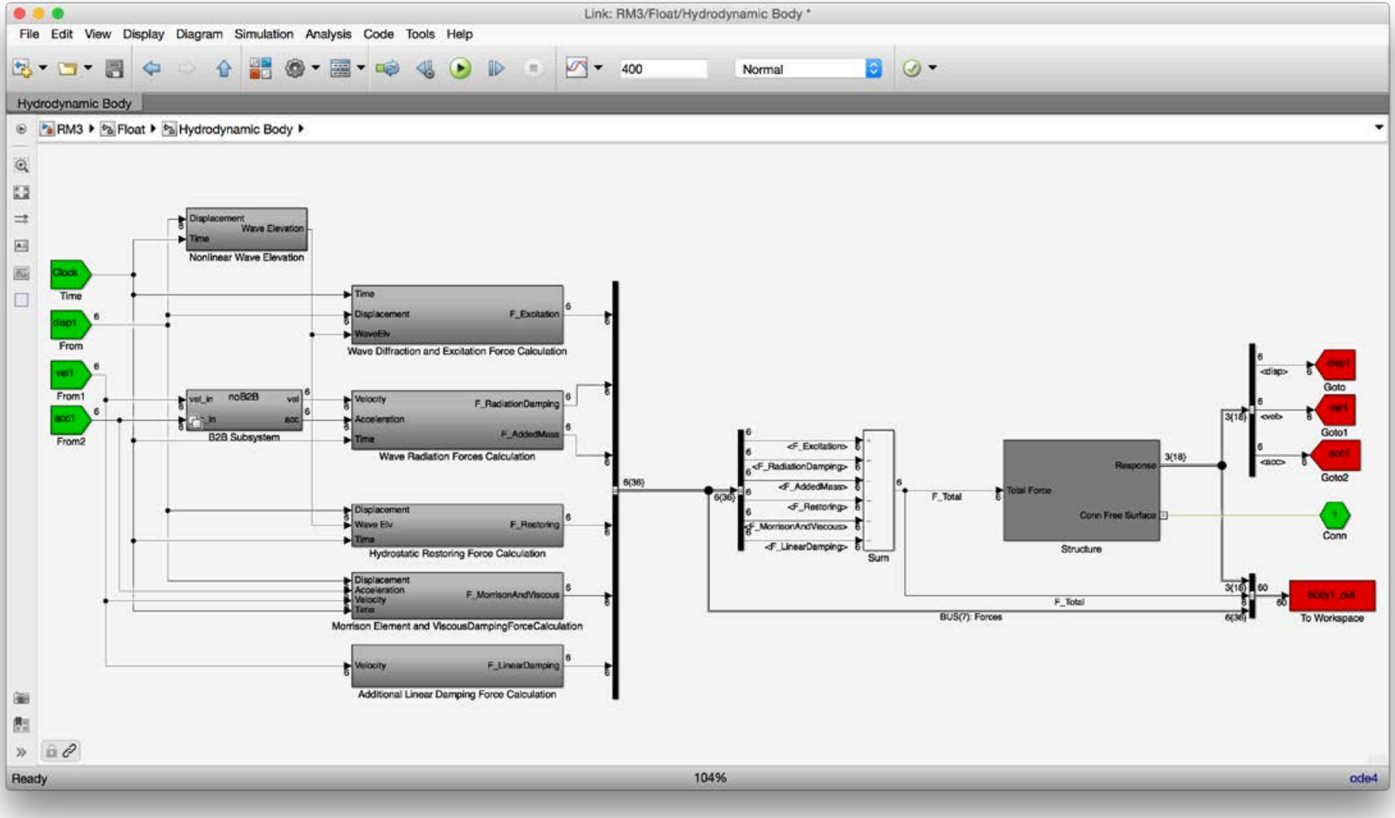
# Hydrodynamic Body Block:

## Wave Radiation Force Calculation Block

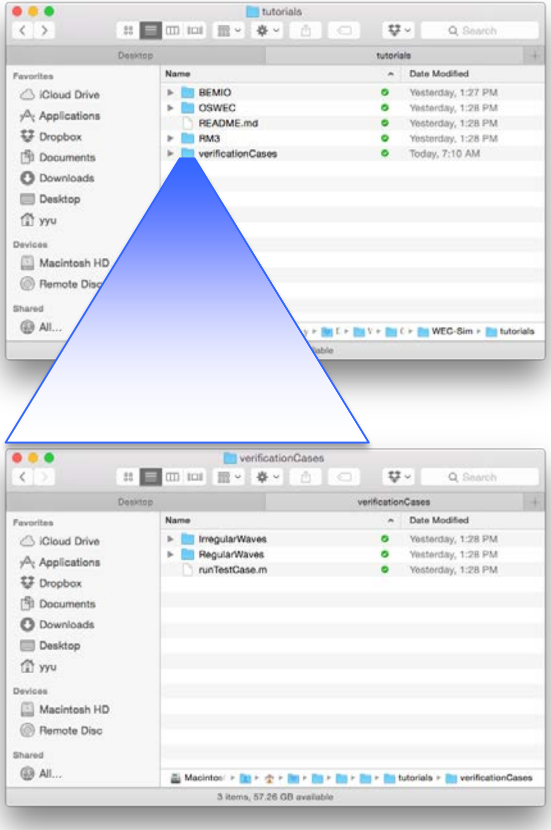


- Added-mass term is a function of acceleration. Therefore, a Transport Delay block is used to break the algebraic loop
- Variable Time-Step (ode45) option is available in WEC-Sim. However, the Convolution Integral block is always calculated using a fixed-time step.

# Body Force Dimension Display



# Verifications Cases



tutorials

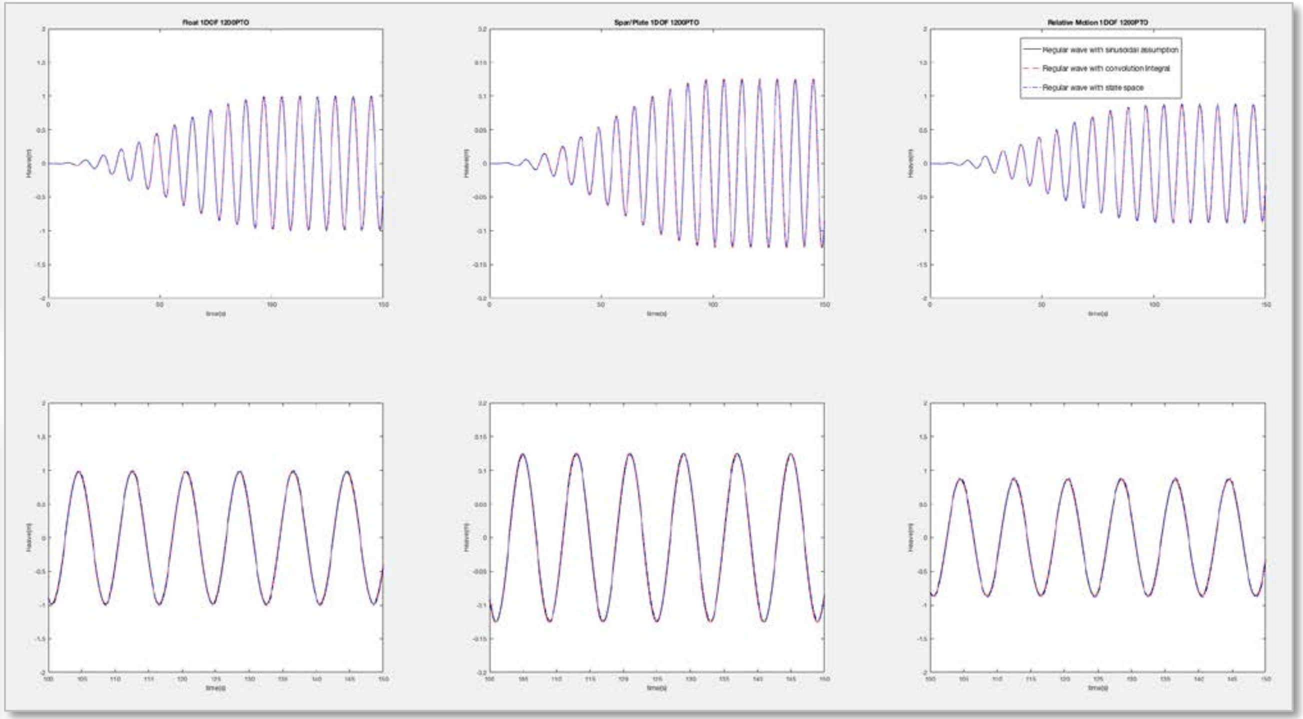
| Name | Date Modified |
|---|---|
| BEMIO | Yesterday, 1:27 PM |
| OSWEC | Yesterday, 1:28 PM |
| README.md | Yesterday, 1:28 PM |
| RM3 | Yesterday, 1:28 PM |
| verificationCases | Today, 7:10 AM |

WEC-Sim ▸ tutorials

verificationCases

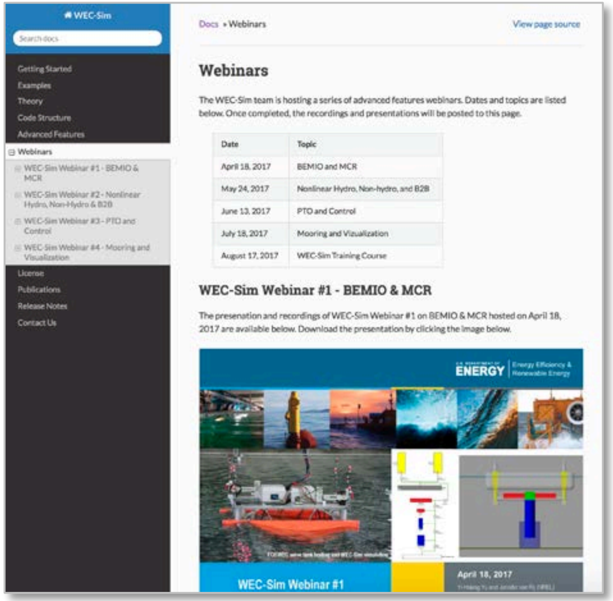
| Name | Date Modified |
|---|---|
| IrregularWaves | Yesterday, 1:28 PM |
| RegularWaves | Yesterday, 1:28 PM |
| runTestCase.m | Yesterday, 1:28 PM |

Macintosh ▸ tutorials ▸ verificationCases



# Thank you!

***All the webinar materials and recordings are available online***:
[http://wec-sim.github.io/WEC-Sim/webinars.html](http://wec-sim.github.io/WEC-Sim/webinars.html)


WEC-Sim
Search docs
Getting Started
Examples
Theory
Code Structure
Advanced Features
Webinars
WEC-Sim Webinar #1 - BEMIO & MCR
WEC-Sim Webinar #2 - Nonlinear Hydro, Non-Hydro & B2B
WEC-Sim Webinar #3 - PTO and Control
WEC-Sim Webinar #4 - Mooring and Visualization
License
Publications
Release Notes
Contact Us
Docs » Webinars
View page source

## Webinars

The WEC-Sim team is hosting a series of advanced features webinars. Dates and topics are listed below. Once completed, the recordings and presentations will be posted to this page.

| Date | Topic |
|---|---|
| April 18, 2017 | BEMIO and MCR |
| May 24, 2017 | Nonlinear Hydro, Non-hydro, and B2B |
| June 13, 2017 | PTO and Control |
| July 18, 2017 | Mooring and Vizualization |
| August 17, 2017 | WEC-Sim Training Course |

### WEC-Sim Webinar #1 - BEMIO & MCR

The presenation and recordings of WEC-Sim Webinar #1 on BEMIO & MCR hosted on April 18, 2017 are available below. Download the presentation by clicking the image below.

U.S. DEPARTMENT OF ENERGY
Energy Efficiency & Renewable Energy
WEC-Sim Webinar #1
April 18, 2017

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.