SESAM USER MANUAL

WAMOD

Wave Modeller

Valid from Wamod version Valid from program version 2.0





Sesam User Manual

Wamod

Date:

Valid from Wamod version Valid from program version 2.0

Prepared by: Digital Solutions at DNV

E-mail support: software.support@dnv.com

E-mail sales: digital@dnv.com

©DNV AS. All rights reserved

This publication or parts thereof may not be reproduced or transmitted in any form or by any means, including copying or recording, without the prior written consent of DNV AS.

Table of contents

1 INTRODUCTION . . . 2
1.1 Introduction to Wamod . . . 2
1.2 Wamod in the Sesam System . . . 2
1.3 How to read this Manual . . . 3
1.4 Status List . . . 3
1.5 Wamod Extensions . . . 3
2 Theory of Higher Order Spectral Method . . . 4
2.1 Governing equations . . . 4
2.2 Solution procedure . . . 4
3 Numerical implementation . . . 6
3.1 Initialization . . . 7
3.1.1 Linear initialization . . . 7
3.1.2 Nonlinear initialization . . . 7
3.2 Wave breaking model . . . 8
3.2.1 Breaking criterion . . . 8
3.2.2 Wave breaking model = 5 . . . 9
3.2.3 Wave breaking model = 6 . . . 9
4 Execution of Wamod . . . 10
4.1 How to run Wamod . . . 10
4.2 Input parameters . . . 10
4.2.1 Solver settings . . . 10
4.2.2 Initial condition and initialization . . . 12
4.2.3 Physical constants . . . 12
4.2.4 Computational domain . . . 12
4.2.5 Initial condition and initialization . . . 13
4.2.6 Wave breaking option . . . 15
4.2.7 Output options . . . 16
4.3 SWD format . . . 16
4.4 Program Limitations . . . 17
5 Running Wamod from Sesam Manager . . . 18
References . . . 22

1 INTRODUCTION

1.1 Introduction to Wamod

In many ocean engineering problems, ocean waves are an essential input for calculation of e.g. loads and responses of marine structures. Traditionally, when it comes to irregular waves, the shipping and offshore industries have typically been limited to the analysis of linear – or at best – second-order irregular wave fields. It is well-known that neither linear nor second-order theories are able to describe the shape and kinematics of very steep waves realistically. In addition, these theories fail to describe the important nonlinear dynamics among free waves that may have an important impact on the statistical properties of the waves, for example with respect to occurrence of so-called rogue or freak waves, see [1] for an overview.

For a more realistic description of waves and their associated effects on loads and responses, a nonlinear description may be applied. One challenge in this respect is that in contrast to linear and second-order theories where analytical solutions are available, higher order nonlinear waves must be described through numerical simulations of the basic hydrodynamic equations. The Higher Order Spectral Method (HOSM), commonly also abbreviated HOS, first proposed independently by [2] and [3], is a numerical method for solving the fully nonlinear equations for waves on the surface of an ideal fluid. Its main advantage compared to many other methods is the numerical efficiency, enabling the description of long-crested as well as short crested sea states in reasonable computational time. In addition, the HOSM formulation enables an efficient calculation of the water particle kinematics below the surface. This is essential input to tools such as Wasim for the calculation of wave loads and responses.

The output file of Wamod is in SWD (Spectral Wave Data) format and follows strictly with the definition given by https://github.com/SpectralWaveData. C++/Fortran/Python interfaces are available for SWD format and example files can be found on this Github page.

Wasim is a Rankine panel program for simulating wave induced responses of marine structures. This time domain hydrodynamic analysis module under Sesam provides the interface to this SWD format. The incident wave kinematics and Froude-Krylov forces on the structures will be calculated based on the wave kinematics information output by Wamod and stored through SWD format. All the wave models in Wamod have been successfully applied in Wasim. More details can be found in Wasim User Manual.

1.2 Wamod in the Sesam System

Wamod is an integrated part of the Sesam suite of programs, with interfaces to the hydrodynamic analysis module Wasim.

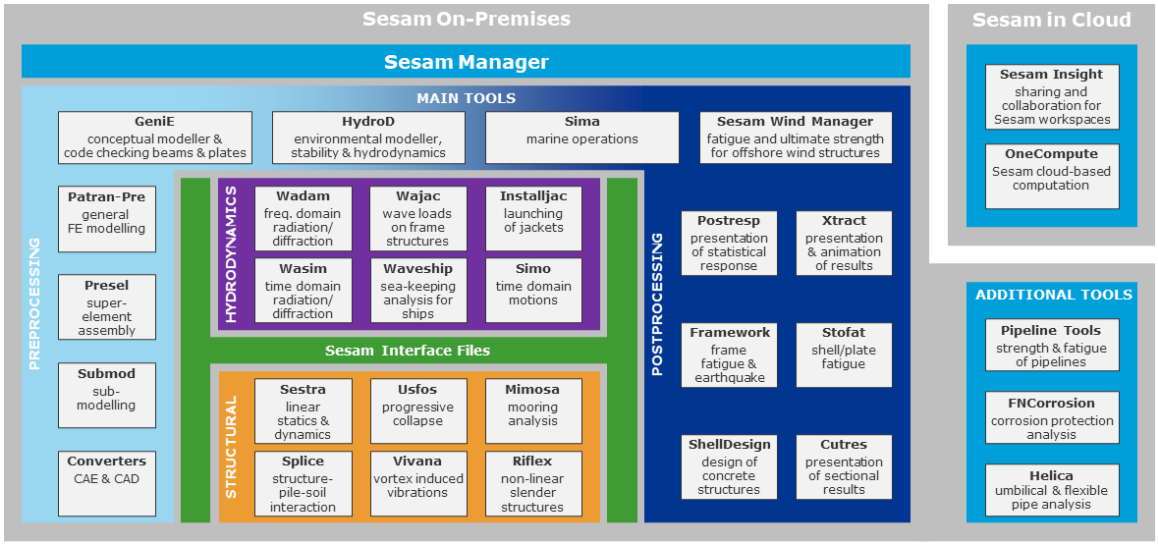


Figure 1.1: Sesam overview

1.3 How to read this Manual

The user manual is organized as follows: Section 2 presents the theoretical background of Wamod and numerical implementation in Wamod. An overview of all the relevant input parameters and options needed for running the program is given in Section 4. Section 5 illustrates how to run Wamod through Sesam Manager instead of command line.

A good approach to learn Wamod is to start by studying Section 2, Section 4 and then proceed by following the examples included in the installation package of Wamod. The input files to the examples are provided as a part of the installation and can be found in the UserExamples directory under the program installation package. For a more extensive Sesam analysis, Section 5 can be referred to on how to define a Wamod template and thus set up Wamod activity in Sesam Manager.

1.4 Status List

There exists for Wamod (as for all other Sesam programs) a Status List providing additional information. This may be:

- Reasons for update (new version)
- New features
- Errors found and corrected
- Etc.

The status lists can be accessed on the site: `https://sesam.dnv.com`.

1.5 Wamod Extensions

NLIN Non-linear wave simulation by use of Higher Order Spectral Method. An extra license is required for those relevant features, specified in Section 4.2.

2 Theory of Higher Order Spectral Method

This chapter describes the theory and numerical implementation of Wamod.

2.1 Governing equations

The Higher Order Spectral Method (HOSM) is based on the formulation of the water wave problem (the Laplace equation with kinematic and dynamic boundary conditions at the free surface) in the surface variables: the surface elevation $\eta(x,y,t)$ and the surface potential $\psi(x,y,t) = \phi(x,y,z=\eta(x,y,t),t)$, where $\phi(x,y,z,t)$ is the velocity potential in the fluid domain. This formulation of the water wave basic equations was first presented by [4], who showed that they constitute a Hamiltonian system, with η and ψ as Hamiltonian variables, in the form

$$\frac{\partial \eta}{\partial t} = w_s\left(1+|\nabla\eta|^2\right) - \nabla\psi\cdot\nabla\eta, \tag{2.1a}$$

$$\frac{\partial \psi}{\partial t} = -g\eta - \frac{1}{2}|\nabla\psi|^2 + \frac{w_s^2}{2}\left(1+|\nabla\eta|^2\right), \tag{2.1b}$$

where $w_s = \left.\frac{\partial\phi}{\partial z}\right|_{z=\eta}$ is the vertical surface velocity and where $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$ is the horizontal gradient operator.

2.2 Solution procedure

The system Equation (2.1) can be solved numerically using standard solvers for ordinary differential equations (ODEs). This requires, however, that one can express the vertical surface velocity w_s in terms of η and ψ. An efficient iterative spectral method for evaluating w_s in terms of η and ψ was proposed by [2] and [3], and has later been known as the HOS method. It has later been shown that HOSM alternatively can be derived and formulated in terms of the Dirichlet–Neumann operator [5, 6]. Here we will give a brief summary of the original approach due to [3].

We first assume that the full velocity potential $\phi(x,y,z,t)$ can be expanded in a perturbation series:

$$\phi(x,y,z,t) = \sum_{m=1}^{\infty}\phi^{(m)}(x,y,z,t), \tag{2.2}$$

where m reflects the order of each term. Further, assuming that each $\phi^{(m)}$ can be expanded in a Taylor series around the quiescent water surface $z=0$ gives

$$\psi(x,y,t) = \sum_{m=1}^{\infty}\sum_{n=0}^{\infty}\frac{\eta^n}{n!}\frac{\partial^n\phi^{(m)}}{\partial z^n}(x,y,0,t). \tag{2.3}$$

Further, by using the following series law (see e.g. [7])

$$\sum_{m=0}^{\infty}\sum_{n=0}^{\infty}a_m b_n = \sum_{m=0}^{\infty}\sum_{n=0}^{m}a_m b_{m-n}, \tag{2.4}$$

the summations can be rewritten into the form

$$\psi(x,y,t) = \sum_{m=1}^{\infty}\left[\phi^{(m)}(x,y,0,t) + \sum_{n=1}^{m-1}\frac{\eta^n}{n!}\frac{\partial^n\phi^{(m-n)}}{\partial z^n}(x,y,0,t)\right]. \tag{2.5}$$

It is easy to see that for each m, all terms inside the square brackets in Equation (2.5) are of the same order. Then, solving Equation (2.5) at each order gives the following iterative solution for $\phi(x,y,0,t)$:

$$\phi^{(1)}(x,y,0,t) = \psi(x,y,t), \tag{2.6a}$$

$$\phi^{(m)}(x,y,0,t) = -\sum_{n=1}^{m-1}\frac{\eta^n}{n!}\frac{\partial^n\phi^{(m-n)}}{\partial z^n}(x,y,0,t), \quad m\geq 2. \tag{2.6b}$$

We now assume the following Fourier representation of $\phi^{(m)}$

$$\phi^{(m)}(x,y,z,t) = \mathcal{F}^{-1}\left\{\hat{\phi}^{(m)}(k_x,k_y,t)\frac{\cosh k(z+h)}{\cosh kh}\right\}, \tag{2.7}$$

where $\mathcal{F}\{\cdot\}$ and $\mathcal{F}^{-1}\{\cdot\}$ denote the Fourier-transform and its inverse from the horizontal plane $\boldsymbol{x} = (x,y)$ to the wavenumber plane $\boldsymbol{k} = (k_x,k_y)$, and where $k = |\boldsymbol{k}|$. It is then easy to show that

$$\frac{\partial^n \phi^{(m)}}{\partial z^n}(x,y,0,t) = \mathcal{F}^{-1}\left\{\hat{\phi}^{(m)}E_n\right\}, \tag{2.8}$$

where

$$E_n = \begin{cases} k^n & \text{for } n \text{ even,} \\ k^n \tanh(kh) & \text{for } n \text{ odd.} \end{cases} \tag{2.9}$$

Using the above, we see that the three first orders of $\phi^{(m)}$ are

$$\phi^{(1)} = \psi, \tag{2.10a}$$

$$\phi^{(2)} = -\eta\mathcal{F}^{-1}\{k\tanh(kh)\mathcal{F}\{\psi\}\}, \tag{2.10b}$$

$$\phi^{(3)} = \eta\mathcal{F}^{-1}\left\{k\tanh(kh)\mathcal{F}\left\{\eta\mathcal{F}^{-1}\{k\tanh(kh)\mathcal{F}\{\psi\}\}\right\}\right\} - \frac{\eta^2}{2}\mathcal{F}^{-1}\left\{k^2\mathcal{F}\{\psi\}\right\}. \tag{2.10c}$$

In the same way we can express the vertical surface velocity w_s in the form

$$w_s(x,y,t) = \sum_{m=1}^{\infty}\sum_{n=0}^{m-1}\frac{\eta^n}{n!}\frac{\partial^{n+1}\phi^{(m-n)}}{\partial z^{n+1}}(x,y,0,t). \tag{2.11}$$

Hence, the surface velocity at order m takes the form

$$w_s^{(m)}(x,y,t) = \sum_{n=0}^{m-1}\frac{\eta^n}{n!}\frac{\partial^{n+1}\phi^{(m-n)}}{\partial z^{n+1}}(x,y,0,t) = \sum_{n=0}^{m-1}\frac{\eta^n}{n!}\mathcal{F}^{-1}\left\{E_{n+1}\mathcal{F}\left\{\phi^{(m-n)}\right\}\right\}. \tag{2.12}$$

It is clear that $\phi^{(m)}$ and $w_s^{(m)}$ can be calculated iteratively.

3 Numerical implementation

For numerical implementation of the procedures explained above, discretization of the physical space (x, y) and Fourier space (k_x, k_y) are introduced. We resolve a rectangular domain in the (x, y) plane discretized by $n_x \times n_y$ points so that

$$(x_j, y_k) = (j\Delta x, k\Delta y), \quad \text{for} \quad j = 0, \cdots, n_x - 1, \quad k = 0, \cdots, n_y - 1, \tag{3.1}$$

for some chosen grid spacing Δx and Δy. It is assumed that in the grid points, η and ψ can be represented in terms of an inverse discrete Fourier transform

$$\eta_{j,k} = \eta(j\Delta x, k\Delta y) = \sum_{u=0}^{n_x-1} \sum_{v=0}^{n_y-1} \hat{\eta}_{u,v} \mathrm{e}^{2\pi \mathrm{i}(\frac{ju}{n_x} + \frac{kv}{n_y})}, \tag{3.2}$$

with the forward transform

$$\hat{\eta}_{u,v} = \frac{1}{n_x n_y} \sum_{j=0}^{n_x-1} \sum_{k=0}^{n_y-1} \eta_{j,k} \mathrm{e}^{-2\pi \mathrm{i}(\frac{uj}{n_x} + \frac{vk}{n_y})}. \tag{3.3}$$

These discrete Fourier transforms are calculated efficiently by FFT, where the Intel Fortran Math Kernel Library (MKL) FFT routines are used.

Note that Equation (3.2) implies periodic boundary conditions for η and ψ, e.g. $\eta(0, y_k) = \eta(L_x, y_k)$ and $\eta(x_j, 0) = \eta(x_j, L_y)$, where $L_x = n_x \Delta x$ and $L_y = n_y \Delta y$. That is, we solve the equations in a rectangular domain with periodic boundary conditions in both dimensions. Furthermore, due to the fact that η and ψ are real, we know that $\hat{\eta}_{u,v} = \hat{\eta}^*_{n_x-u, n_y-v}$ where both dimensions are assumed periodic so that $\hat{\eta}_{n_x,v} = \hat{\eta}_{0,v}$ and $\hat{\eta}_{u,n_y} = \hat{\eta}_{u,0}$. Here $*$ denotes complex conjugate.

The points in wavenumber space corresponding to the Fourier-coefficients $\hat{\eta}_{u,v}$ are given by

$$k_{xu} = \left(u - \frac{n_x}{2} + 1\right) \Delta k_x, \quad \text{for} \quad u = 0, \cdots, n_x - 1, \tag{3.4a}$$

$$k_{y_v} = \left(v - \frac{n_y}{2} + 1\right) \Delta k_y, \quad \text{for} \quad v = 0, \cdots, n_y - 1, \tag{3.4b}$$

where

$$\Delta k_x = \frac{2\pi}{n_x \Delta x} = \frac{2\pi}{L_x}, \qquad \Delta k_y = \frac{2\pi}{n_y \Delta y} = \frac{2\pi}{L_y}. \tag{3.5}$$

That is, the resolved domain in wavenumber space is approximately $k_x \in [-\frac{\pi}{\Delta x}, \frac{\pi}{\Delta x}]$ and $k_y \in [-\frac{\pi}{\Delta y}, \frac{\pi}{\Delta y}]$. Note that if n_x or n_y are odd Equation (3.4a) is replaced by

$$k_{xu} = \left(u - \frac{n_x - 1}{2}\right) \Delta k_x, \quad \text{for} \quad u = 0, \cdots, n_x - 1, \tag{3.6}$$

and correspondingly for Equation (3.4b).

The water particle kinematics within the Wamod formulation is calculated using the so-called H_2-operator by [8].

Note

The computational grid in HOSM is controlled by the user through namelist parameters nx, kxmax and xdom, and correspondingly in case of 2-dimensional simulations, ny, kymax and ydom. See more details in Section 4.2.4.

3.1 Initialization

As the initial condition Wamod needs the initial surface elevation $\eta(x, y, t = 0)$ and initial surface potential $\psi(x, y, t = 0)$, which are initialized according to the chosen wave spectrum (check `itype` in Section 4.2.5.), and choice of random phases/amplitudes (check `ramp_seed` and `ramp_amp` in Section 4.2.5.). One can show that η and ψ can be expressed in terms of a common complex variable $a(\boldsymbol{k})$ so that

$$\begin{aligned}\eta(x, y) &= \mathcal{F}^{-1}\left\{\sqrt{\frac{\omega(\boldsymbol{k})}{2g}}a(\boldsymbol{k})\right\} + *,\\ \psi(x, y) &= -\mathrm{i}\mathcal{F}^{-1}\left\{\sqrt{\frac{g}{2\omega(\boldsymbol{k})}}a(\boldsymbol{k})\right\} + *,\end{aligned} \tag{3.7}$$

where $\omega(\boldsymbol{k}) = \sqrt{g|\boldsymbol{k}|\tanh(|\boldsymbol{k}|h)}$ is the linear dispersion relation, and $\mathcal{F}^{-1}$ denotes the inverse Fourier-transform. In the general nonlinear case the variable $a(\boldsymbol{k})$ is a combination free and bound waves.

3.1.1 Linear initialization

Under the linear approximation one can write $a(\boldsymbol{k}) = b(\boldsymbol{k})$, where $b(\boldsymbol{k})$ is the part of $a(\boldsymbol{k})$ representing free waves only. Typically, one wants to initialize according to a given wave spectrum $S(\boldsymbol{k})$ that represents the energy of the free waves. In this case one would choose

$$b_{i,j} = b(\boldsymbol{k}_{i,j}) = \sqrt{\frac{g}{\omega(\boldsymbol{k}_{i,j})}S(\boldsymbol{k}_{i,j})}A_{i,j}\mathrm{e}^{i\theta_{i,j}},$$

where $\theta_{i,j}$ are random phases drawn from a uniform distribution on $[0, 2\pi]$, and $A_{i,j}$ are random amplitudes drawn from a Rayleigh distribution with parameter $\sigma = 1/\sqrt{2}$. Note that this corresponds to each wave component $b_{i,j}$ being complex normally distributed with $E[|b_{i,j}|] = \sqrt{\frac{g}{\omega_{i,j}}S(\boldsymbol{k}_{i,j})}$.

3.1.2 Nonlinear initialization

For a fully consistent initialization, η and ψ should be chosen according to their nonlinear relationship that represents bound wave components correctly. Given a set of free (linear) wave modes $b(\boldsymbol{k})$, one can represent $a(\boldsymbol{k})$ in terms of an integral power series in $b(\boldsymbol{k})$. This power series can be obtained through the so-called canonical transformation that expresses bound waves in terms of free waves obtained during the derivation of the Zakharov equation. This canonical transformation up to fourth-order nonlinearity (corresponding to HOSM order $M = 4$) was derived by [17], and is to second order given as

$$\begin{aligned}a(\boldsymbol{k}) = b(\boldsymbol{k}) &+ \int A^{(1)}(\boldsymbol{k}, \boldsymbol{k}_1, \boldsymbol{k}_2)b(\boldsymbol{k}_1)b(\boldsymbol{k}_2)\delta(\boldsymbol{k} - \boldsymbol{k}_1 - \boldsymbol{k}_2)\,\mathrm{d}\boldsymbol{k}_1\,\mathrm{d}\boldsymbol{k}_2\\ &+ \int A^{(2)}(\boldsymbol{k}, \boldsymbol{k}_1, \boldsymbol{k}_2)b^*(\boldsymbol{k}_1)b(\boldsymbol{k}_2)\delta(\boldsymbol{k} + \boldsymbol{k}_1 - \boldsymbol{k}_2)\,\mathrm{d}\boldsymbol{k}_1\,\mathrm{d}\boldsymbol{k}_2\\ &+ \int A^{(3)}(\boldsymbol{k}, \boldsymbol{k}_1, \boldsymbol{k}_2)b^*(\boldsymbol{k}_1)b^*(\boldsymbol{k}_2)\delta(\boldsymbol{k} + \boldsymbol{k}_1 + \boldsymbol{k}_2)\,\mathrm{d}\boldsymbol{k}_1\,\mathrm{d}\boldsymbol{k}_2 + \cdots.\end{aligned} \tag{3.8}$$

The kernel functions appearing in Equation (3.8) can be found in [17]. As described above the variable $b(\boldsymbol{k})$ in Equation (3.8) represents the free-wave components of the wave field and can typically be chosen as desired, e.g. as a spectrum of independent wave-components, from which η and ψ can be found by using Equation (3.7) and Equation (3.8).

3.2 Wave breaking model

The HOSM equations are energy preserving (and Hamiltonian). Naturally they do not contain any mechanism for dissipation and wave breaking. Hence, for very steep waves where wave breaking plays an important role the solution may be less realistic, or the simulations may even break down due to accumulation of energy at high wave numbers. This is a typical limitation of all potential flow codes. However, some effects of wave breaking may be included by applying pragmatic models for the wave breaking. While such wave breaking models cannot describe the complex physics of the actual wave breaking process, it may in many cases be a quite good model for the overall effects of the wave breaking in the wave field.

The local wave breaking model is implemented in Wamod. It is based on detecting the individual waves that exceed some breaking criteria. Then, the breaking terms are applied locally to the individual breaking waves. The modelling of the actual breaking process implements the wave breaking model described in [10] and [11] by setting `breaking = 5` or `breaking = 6` (see more details in 4.2.6).

Such wave breaking model can be applied to avoid unphysical waves. However it is not a way to describe the actual wave breaking accurately (impossible in this kind of code), where more advanced CFD codes should be used instead.

3.2.1 Breaking criterion

The local breaking models all use the wave-breaking onset criterion described by [12] which uses the ratio of the water particle velocity under the wave crest u_x and the local crest velocity c_x. That is, a wave is considered to be breaking when

$$B_x = u_x/c_x > \beta \tag{3.9}$$

for some threshold β. It was reported in [12] that the onset of breaking was found to occur when $0.85 < B_x < 0.86$, but a somewhat wider range $0.75 < B_x < 0.95$ has also been reported in the literature (see e.g. [13] and references therein).

As suggested in [14], the local crest velocity can be estimated from the local wave number $k(x,t)$ which can be calculated from the Hilbert-transform of the surface elevation $\tilde{\eta} = H(\eta)$, i.e.

$$c_x(x,t) = \sqrt{\frac{g\tanh\left(k(x,t)h\right)}{k(x,t)}},$$

where

$$k(x,t) = \frac{1}{\tilde{\eta}^2+\eta^2}\left(\eta\frac{\partial\tilde{\eta}}{\partial x} + \tilde{\eta}\frac{\partial\eta}{\partial x}\right).$$

Here, $\tilde{\eta}(x)$ is the Hilbert-transform in space, which due to the periodic computational domain can easily be calculated using FFT.

The particle velocity at the crests is defined as

$$u_x = \left.\frac{\partial\phi}{\partial x}\right|_{z=\eta} = \frac{\partial\psi}{\partial x} - \left.\frac{\partial\phi}{\partial z}\right|_{z=\eta}\frac{\partial\eta}{\partial x}.$$

Since $\partial\eta/\partial x = 0$ at a crest, $u_x = \partial\psi/\partial x$ can be calculated very efficiently in Wamod using the surface potential only.

Hence, at each crest, defined as the maximum within each zero-crossing region, the breaking criterion B_x is evaluated and a wave is "flagged" as breaking if $B_x > \beta$, where β is the wave breaking threshold that can be set in the input file through the parameter of $UCratio$ if `breaking = 5` or is automatically decided based on the input Hs value if `breaking = 6`.

3.2.2 Wave breaking model = 5

This wave breaking model implements the eddy-viscosity model described in [10] and [11], and applies the modified HOSM equations in the following form

$$\begin{aligned}\frac{\partial \eta}{\partial t} &= w_s\left(1+|\nabla\eta|^2\right) - \nabla\psi\cdot\nabla\eta + 2\nu_{eddy}\nabla^2\eta,\\ \frac{\partial \psi}{\partial t} &= -g\eta - \frac{1}{2}|\nabla\psi|^2 + \frac{w_s^2}{2}\left(1+|\nabla\eta|^2\right) + 2\nu_{eddy}\nabla^2\psi,\end{aligned} \tag{3.10}$$

where $\nu_{eddy} = 0$ for waves not satisfying the breaking criterion Equation (3.9), and $\nu_{eddy} = \alpha H_{br} L_{br}/T_{br}$ for waves satisfying the breaking criterion. H_{br}, L_{br} and T_{br} are properties of the breaking wave (see equations (12) - (14) in [11] for details) and α is a constant that has a constant value 0.025 (see [10]) but can be set through parameter $alpha_br5$.

A ramp-function (truncated cosine) is used for a smooth transition between breaking (non-zero ν_{eddy}) and non-breaking (zero ν_{eddy}) regions of the wave-train.

Hence, the procedure of the wave breaking model can be summarized as follows, for a given time t:

- Identify all individual waves in the domain by considering zero-crossings over the entire computational domain x.
- For each individual wave, evaluate if the wave is breaking according to the breaking criterion Equation (3.9).
- For each breaking wave, calculate $\nu_{eddy} = \alpha H_{br} L_{br}/T_{br}$.
- Use ramp-functions and take periodicity of the domain into account to construct a smooth and periodic function $\nu_{eddy}(x)$ that is multiplied with the diffusion terms in Equation (3.10).

While the wave breaking model (including the breaking-onset criterion) is formulated for unidirectional waves, it is also implemented for short-crested waves in Wamod. In this case the breaking procedure is applied independently for each value of y in the grid. This is tested, and seems to work well as long as the main direction of wave propagation is along the x-axis. Thus, it cannot be used for crossing sea states, for example.

This wave breaking mode can only be used with `solver` as HOSM. For more control and in proper studies this wave breaking model should be used together with some investigation of breaking parameters $UCratio$ and/or $alpha_br5$, ideally with some verification against other sources (e.g. experimental data) for the sea states of interest.

3.2.3 Wave breaking model = 6

This is the same as `breaking = 5`, but with automatic selection of breaking tuning parameters, based on sea state properties. Such automatic tuning on parameters is only performed for deep-water waves, so validity for shallow-water is unclear.`breaking = 6` model can only be used with JONSWAP wave spectrum (`itype` = 1) or Torsethaugen wave spectrum (`itype` = 4).

The Runge_Kutta Fehlberg fourth-fifth order solver is used, which requires a constant time step `dt` as input. The sea state wave steepness, defined as $kp * Hs/2$ will be reported in LIS file. If this value is smaller than a threshold value of 0.1 and at deep water $h < 0$, breaking model is usually not necessary thus Wamod will set breaking as 0, i.e. no breaking mode. This was set mainly due to consideration of performance. If no waves reach breaking, limit there will be zero difference between simulation with and without breaking. However some extra simulation time due to constantly checking for breaking waves will be added.

4 Execution of Wamod

This section provides information on:

- How to run Wamod
- Input and output files
- Program requirements
- Program limitations

4.1 How to run Wamod

Wamod is executed through command line with a user input file. After specifying an input file, the program is called from the command line by the command `Wamod inputfile`, where the first line of `inputfile` has to be started with &hosm. One example input file has been given as:

```
&hosm    !Mandatory namelist tag
!-------------------------------------------------------------
M        = 5         ! Nonlinear order of HOSM simulation
solver   = 'HOSM'    ! Type of solver
ndim     = 1         ! Number of horizontal dimensions
nx       = 1024      ! Number of grid points in x-direction
tmax     = 1000.0    ! Length of simulation (in seconds).
itype    = 1         ! Jonswap spectrum
gammaJ   = 3.3       ! Parameter gamma in JONSWAP spectrum.
h   = 160.0          ! Water depth
Hs = 13              ! Significant wave height of JONSWAP spectrum.
Tp = 12              ! Peak period of JONSWAP spectrum.
outfile = 'Hosm_ITYPE1.swd' ! Filename of the output file.
sf_nout = 1000       ! Number of outputs of spatial fields
!-------------------------------------------------------------
/
```

4.2 Input parameters

The following sections describe all relevant input parameters to the Wamod program as well as a description of the output. For those input parameters not listed in this document, the default values will be applied.

4.2.1 Solver settings

`solver = 'HOSM'`: the most common solver used in Wamod to simulate a homogeneous wave field in a periodic spatial domain in time. WAMOD_NLIN license is required

In addition to solver of HOSM, Wamod also includes solvers for linear and second-order irregular wave theory [15].

`solver= 'Linear'`: analytical linear solution. Input is very similar to when using the HOSM solver. In fact, HOSM with nonlinear order `M = 1` is equivalent to the linear solver, but using `solver = 'Linear'` is likely slightly more efficient due to the direct use of the analytical solution.

```
&hosm    !Mandatory namelist tag
!-------------------------------------------------------------------
M        = 1                ! Linear analytical simulation
solver   = 'Linear'         ! Type of solver
ndim     = 1                ! Number of horizontal dimensions
nx       = 1024             ! Number of grid points in x-direction
tmax     = 1000.0           ! Length of simulation (in seconds).
itype    = 1                ! Jonswap spectrum
gammaJ   = 3.3              ! Parameter gamma in JONSWAP spectrum.
```

```
h   = 160.0                ! Water depth
Hs  = 13                   ! Significant wave height
Tp  = 12                   ! Peak period of JONSWAP spectrum.
outfile = 'Linear.swd'     ! Filename of the output file.
sf_nout = 1001             ! Number of outputs of spatial fields
!-------------------------------------------------------------------
/
```

`solver='SecondOrder'`: analytical second-order solution [15]. The input for this solver is very similar to that when using the HOSM solver. It is however noted that HOSM solver with `M = 2` is not totally equivalent to `solver = 'SecondOrder'`. Since the second-order solver requires calculation of all second-order interaction coefficients, which for large number of modes (in particular for short-crested simulations) can be very many. WAMOD_NLIN license is required

```
&hosm   !Mandatory namelist tag
!-----------------------------------------------------------------
M       = 2                ! Second order analytical simulation
solver  = 'SecondOrder'    ! Type of solver
ndim    = 1                ! Number of horizontal dimensions
nx      = 1024             ! Number of grid points in x-direction
tmax    = 1000.0           ! Length of simulation (in seconds).
itype   = 1                ! Jonswap spectrum
gammaJ  = 3.3              ! Parameter gamma in JONSWAP spectrum.
h   = 160.0                ! Water depth
Hs  = 13                   ! Significant wave height.
Tp  = 12                   ! Peak period of JONSWAP spectrum.
outfile = 'SecondOrder.swd' ! Filename of the output file.
sf_nout = 1001             ! Number of outputs of spatial fields
!-----------------------------------------------------------------
/
```

Besides the solvers for irregular waves, Wamod provides two regular wave solvers: `solver = 'stokes'`: Fifth order analytical Stokes wave solution. The inputs include the wave height, the wave length as well as the desired order of the Stokes wave solution. WAMOD_NLIN license is required

```
&hosm   !Mandatory namelist tag
!-----------------------------------------------------------------
solver      = 'Stokes'          ! Type of solver
wave_height = 12
wave_length = 100
order       = 5
tmax        = 100.0
h           = 70.0
sf_nout     = 101
outfile     = 'stokes_output.swd'
!-----------------------------------------------------------------
/
```

`solver = 'Fenton'`: The high order numerical solution of [16] for stream function wave. The inputs include the wave height, the wave length and the order of the solution. In this case, `order` is the number of Fourier-components in the solution. WAMOD_NLIN license is required

```
&hosm   !Mandatory namelist tag
!-----------------------------------------------------------------
solver      = 'Fenton'
wave_height = 12
```

```
wave_length = 100
order       = 32
tmax        = 100.0
h           = 70.0
sf_nout     = 101
outfile     = 'fenton_output.swd'
!-----------------------------------------------------------------
/
```

4.2.2 Initial condition and initialization

Parameter	Type	Default	Description
M	integer	3	Nonlinear order of Wamod simulation (If $M \geq 2$, WAMOD_NLIN license is required).
solver	string	'HOSM'	Type of solver: 'HOSM': Nonlinear HOSM solver ($M \geq 2$, WAMOD_NLIN license is required). 'Linear': Linear analytical solution ($M = 1$). 'SecondOrder': Second-order analytical solution ($M = 2$). 'Stokes': Stokes wave solution. 'Fenton': Fenton Stream function wave solution.
relT	float	10^{-3}	Relative error tolerance for ODE solver.
absT	float	10^{-14}	Absolute error tolerance for ODE solver.
wave_height	float	0.2	Regular wave height, only relevant for Stokes and Fenton solver.
wave_length	float	2*π	Regular wave length, only relevant for Stokes and Fenton solver.
order	integer	0	Order of Stokes wave for Stokes wave or number of coefficients in Fenton stream function solver.

4.2.3 Physical constants

Parameter	Type	Default	Description
g	float	9.81	Acceleration of gravity.
h	float	-1.0	Water depth (in meters). Negative value corresponds to infinite depth.

4.2.4 Computational domain

Parameter	Type	Default	Description
ndim	integer	1	Number of horizontal dimensions (1 (long crested) or 2 (short crested)).
nx†	integer	1024	Number of grid points in x-direction (after de-aliasing). Power of two recommended for maximum efficiency.
ny†	integer	1	As above for y-direction. Only relevant if ndim=2.
xdom†	float	-1.0	Size of computational domain in x-direction.
ydom†	float	-1.0	As above for k_y. Only relevant if ndim=2.
kxmax†	float	-8.0	Maximum wavenumber $k_{x_{max}}$ resolved by the grid. A negative value indicate $k_{x_{max}}$ in multiples of k_p. The input value has to be a negative integer.
kymax†	float	-8.0	As above for k_y. Only relevant if ndim=2. The input value has to be a negative integer.
tmax	float	1200.0	Length of simulation (in seconds).

† The parameters (nx, ny), (kxmax, kymax) and (xdom, ydom) are interrelated due to the fact that xdom=pi*nx/kxmax and ydom=pi*ny/kymax, and can therefore not be chosen independently. These parameters can be chosen in the following ways:

kxmax $\neq 0$, xdom < 0: xdom follows from nx and kxmax, i.e. xdom=pi*nx/kxmax.

kxmax $\neq 0$, xdom > 0: nx is adjusted to the smallest power of two that gives the requested domain size xdom, i.e. nx = 2**ceiling(log(xdom*kxmax/pi)/log(2)). Any given value for nx is ignored.

Correspondingly for ny, kymax and ydom.

4.2.5 Initial condition and initialization

Parameter	Type	Default	Description
itype	integer	1	Type of initial condition: itype=1: JONSWAP spectrum. itype=4: Torsethaugen spectrum. itype=5: Two JONSWAP spectra (crossing seas). See below for input parameters corresponding to each of the initial condition types.

`jonswap_fmt`	integer	1	Jonswap wave spectrum transformation format: 1: Use Jonswap-omega-spectrum with correct and consistent dispersion relation considering the effect of finite depth. 2: Use Jonswap-omega-spectrum based on deep water dispersion relation even on finite depth, same as Wamod 1.0 which uses Jonswap-k-spectrum. This option will be applied too if `h` is not set or as -1, i.e. infinite water depth. 3: Correct and consistent transformation of finite depth TMA spectrum. If `TMA` (only available in Wamod 1.0) is set as true,`jonswap_fmt` will be reset as 3 for `itype` = 1 or 5. TMA spectrum or `jonswap_fmt` as 3 is not allowed for `itype` = 4. For infinite water depth,`jonswap_fmt` 1, 2 and 3 are equivalent.

Parameters for `itype=1`

`Tp`†	float	-1.0	Peak period of JONSWAP spectrum.
`Hs`	float	5.5	Significant wave height of JONSWAP spectrum.
`gammaJ`	float	3.3	Parameter γ in JONSWAP spectrum.
`NJ`	integer	4	Directional spreading parameter in the directional distribution $\cos^N(\theta)$ (only relevant if `ndim`=2).
`rand_amp`	logical	True	If false, only random phases (i.e. deterministic amplitudes) are applied in initialization from spectrum.
`rand_seed`	integer	1	Random seed for random number generator (affects the random amplitudes and phases).
`init_secondorder`	logical	False	If true, use second-order theory to initialize η and ψ (see 3.1.2). Only relevant for 'HOSM' solver.

† Only one of `Tp`, `Tz` and `kpJ` should be set in input file. The others two will be calculated accordingly.

Parameters for `itype=4`

`Tp`†	float	-1.0	Peak period of Torsethaugen spectrum.
`Hs`	float	5.5	Significant wave height of Torsethaugen spectrum.
`NJ`	integer	4	Directional spreading parameter in the directional distribution $\cos^N(\theta)$ (only relevant if `ndim`=2).

† Initial condition as a Torsethaugen two-peak spectrum, as described in Appendix A in [9]. This initial condition only accepts `Hs` and `Tp` as input parameters, as well as a directional spreading parameter `NJ` if `ndim = 2`.

Parameters for `itype=5`

`Tp1, Tp2`†	float	-1.0, -1.0	Peak periods of the two JONSWAP spectra.
`Tz1, Tz2`†	float	-1.0, -1.0	Zero-crossing periods of the two JONSWAP spectra.
`kp1, kp2`†	float	-1.0, -1.0	Peak wavenumbers of the two JONSWAP spectra.
`thp1, thp2`	float	-22.5, 22.5	Peak wave directions (in degrees) of the two JONSWAP spectra.
`Hs1, Hs2`	float	2.0, 2.0	Significant wave heights of the two JONSWAP spectra. Note that the significant wave-height of the combined field will be $H_s = \sqrt{H_{s1}^2 + H_{s2}^2}$.
`gamma1, gamma2`	float	3.3, 3.3	Parameters γ for the two JONSWAP spectra.
`N1, N2`	integer	4	Directional spreading parameters in the directional distribution $\cos^N(\theta)$ for the two spectra (only relevant if `ndim=2`).
`rand_amp`	logical	True	If false, only random phases (i.e. deterministic amplitudes) are applied in initialization from spectrum.
`rand_seed`	integer	1	Random seed for random number generator (affects the random amplitudes and phases).

† Only one of `Tp`, `Tz` and `kp` should be set in input file. The others two will be calculated accordingly.

For a given Jonswap wave spectrum, the wave steepness criteria suggested by [9] will be checked in program and a warning messages will be output in LIS file if the criteria has been violated. Wamod will also check whether a Jonswap wave spectrum has been given within a reasonable range suggested by [9]. A warning message will be printed out in LIS file if it is outside the range recommended.

4.2.6 Wave breaking option

Parameter	Type	Default	Description
`breaking`	integer	0	Breaking model: `breaking=0`: No breaking model is applied. `breaking=5`: Breaking model applied `breaking=6`: same as 5 but with automatically decided parameters

UCratio	float	0.85	Threshold for breaking-onset criterion: breaking=5: Input parameter breaking=6: Automatically decided by Wamod
alpha_br5	float	0.025	Constant for breaking-term: breaking=5: Input parameter breaking=6: Automatically decided by Wamod
dt	float	0.0	constant time-step for solver used in breaking wave model 6: breaking=6: around Tp/100

WAMOD_NLIN license is required for wave breaking models.

4.2.7 Output options

Parameter	Type	Default	Description
outfile	string	'output'	Filename of the output file. Wamod will always generate an output file with the given name and an extension of swd (Spectral Wave Data), no matter whether the swd extension has been specified in this parameter or not.
sf_nout	integer	0	Number of outputs of spatial fields (i.e. the solution for all spatial points is written to file at sf_nout equally spaced time intervals).

4.3 SWD format

The binary stream in the SWD file, generated by Wamod, is outlined in the following pseudo code:

```
magic
fmt
shp, amp
prog
date
nid
cid
grav, lscale
nstrip, nsteps, dt
order
select case(shp)
case(1)
        n, dk
        do i = 1, nsteps
           h(0), h(1), ..., h(n)
           ht(0), ht(1), ..., ht(n)
           if amp < 3 then
           c(0), c(1), ..., c(n)
                   ct(0), ct(1), ..., ct(n)
                end if
```

```
            end do
.................
.................
.................
case(6)
            n, d
            do i = 1, n
               amp(i), kw(i), gam(i), phs(n)
            end do
end select
```

where h(),ht(),c(), ct() are the temporal amplitudes for surface elevation and incident velocity potential. These temporal amplitudes are in general complex functions. More detailed descriptions on SWD output format can be found at this link: https://spectral-wave-data.readthedocs.io/en/latest/theory.html#. It is possible to check the contents of a swd file by use of a tool written in Python. More details can be found in this github page https://github.com/SpectralWaveData.

4.4 Program Limitations

Wamod might crash in quite nonlinear sea states, where some build-up of high-frequency energy can occur. Physically this is an indication that wave breaking may play a role in the sea state (wave breaking would have prevented such extremely steep individual waves to appear). This is a limitation of most surface wave models, and requires handling of wave breaking model to avoid very steep waves. One possible solution may be to reduce kxmax (e.g. setting to -6 instead of -8, effectively reducing the highest wavenumber in the simulations) or even with another random seed which may run through. In any case, all such solutions probably require more of the user to ensure robust and good results.

5 Running Wamod from Sesam Manager

Running Wamod is typically a part of a more extensive Sesam analysis, where you may want to utilize the created waves in a hydrodynamic analysis, possibly followed by a structural analysis. In the longer perspective Wamod will be run under control of HydroD, but until that is implemented it will be convenient to include Wamod as an activity within a Sesam Manager job. This will also make it possible to include it in e.g. loop excutions where some of the parameters can be automatically modified by using javascript, to include it in analysis templates, include its input and output in backup files for the project or other useful features that Sesam Manager can provide. Since Wamod is a new product there is no built-in controller for it in Sesam Manager, but you can make a controller for Wamod by using the General controller in Sesam Manager.

Open Sesam Manager, create a new job and insert a General activity into the job by drag and drop as shown below.

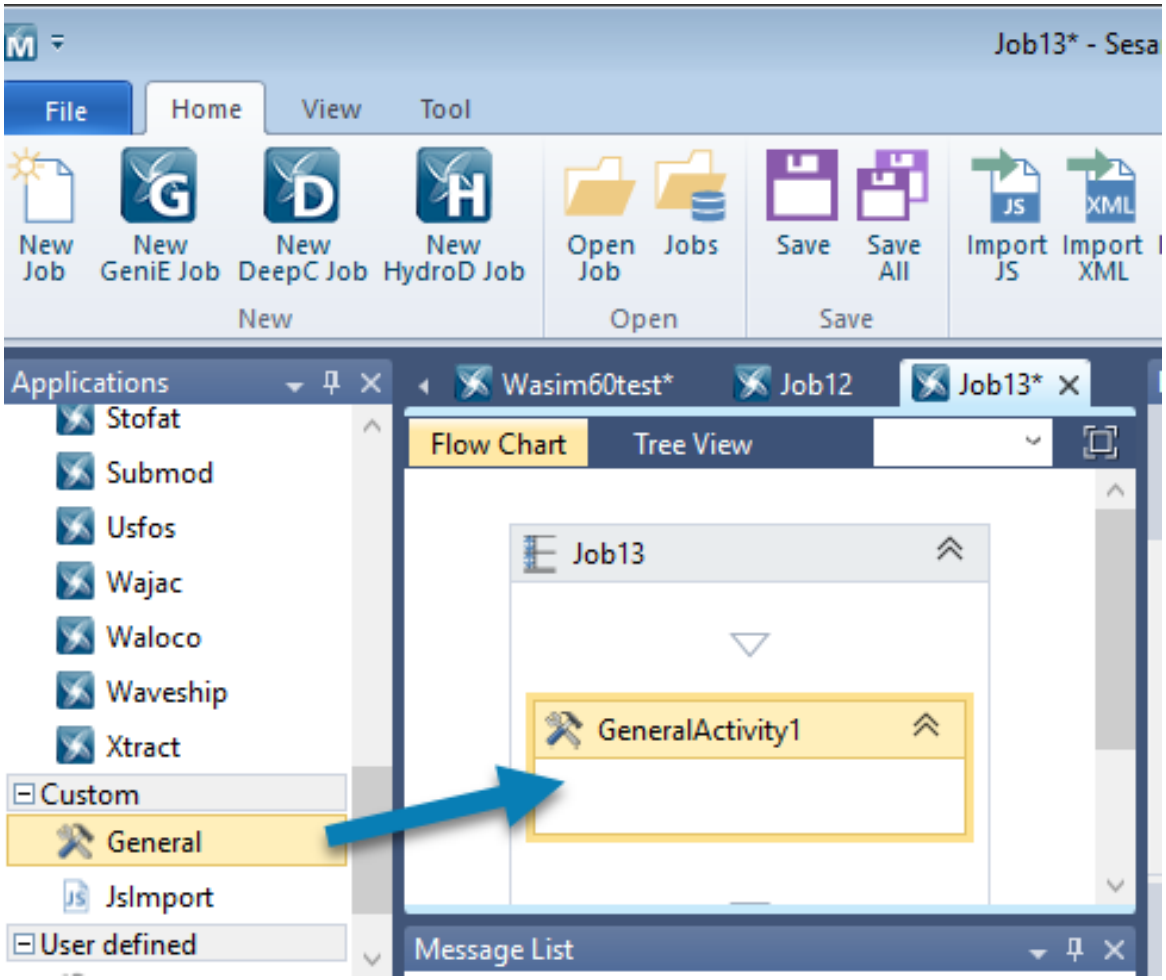


Figure 5.1: Insert a General Activity

Select the general activity and press the Find button in the properties window.

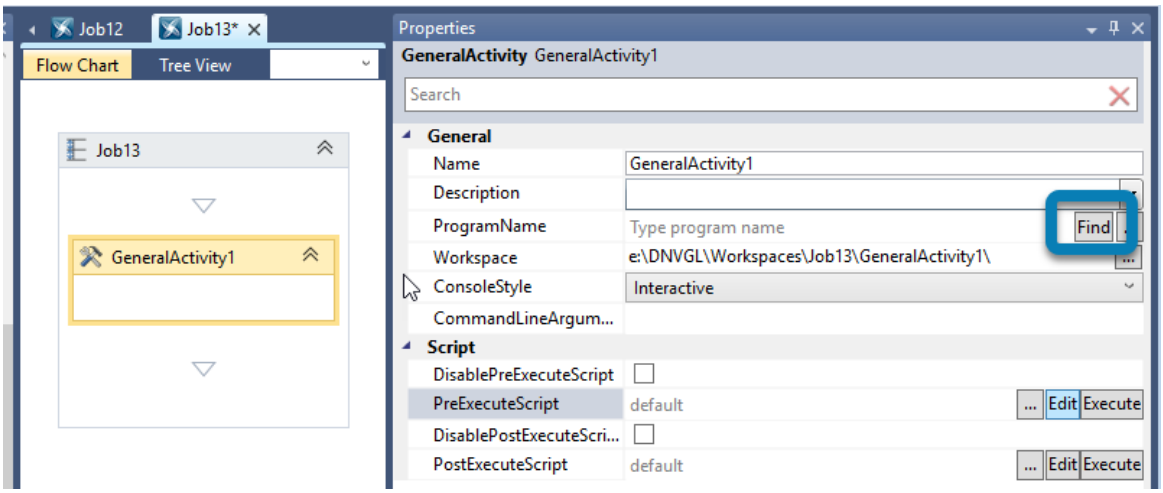


Figure 5.2: Select General Activity

Select Wamod in the list that appears and press OK:

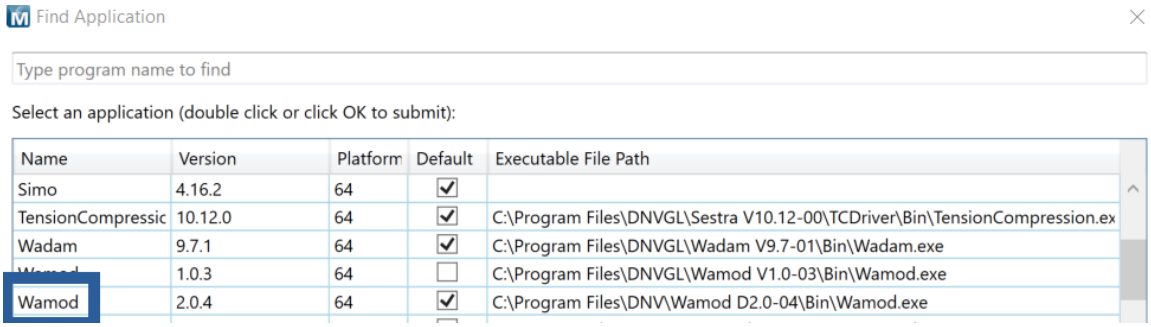


Figure 5.3: Find Wamod in the List

Now select the Tool tab and press the Save as activity template button. Now you may specify the name of the template and define the Pre and PostExecuteScripts that you want to apply. These scripts may be deactivated or modified when the template is used in a specific analysis. In the example shown below the template is given the name Wamod, the PreExecuteScript will copy *.inp files from the specified input folder of the Sesam Manager job to the Wamod workspace folder. The PostExecuteScript will copy the created swd file to the _repository folder.

PreExecuteScript:

Copy(this.InputFileLocation + "*.inp", CurrentActivity.Workspace);

PostExecuteScript:

Copy(CurrentActivity.Workspace + "*.swd","_repository", CurrentActivity.StartExecutionTime);

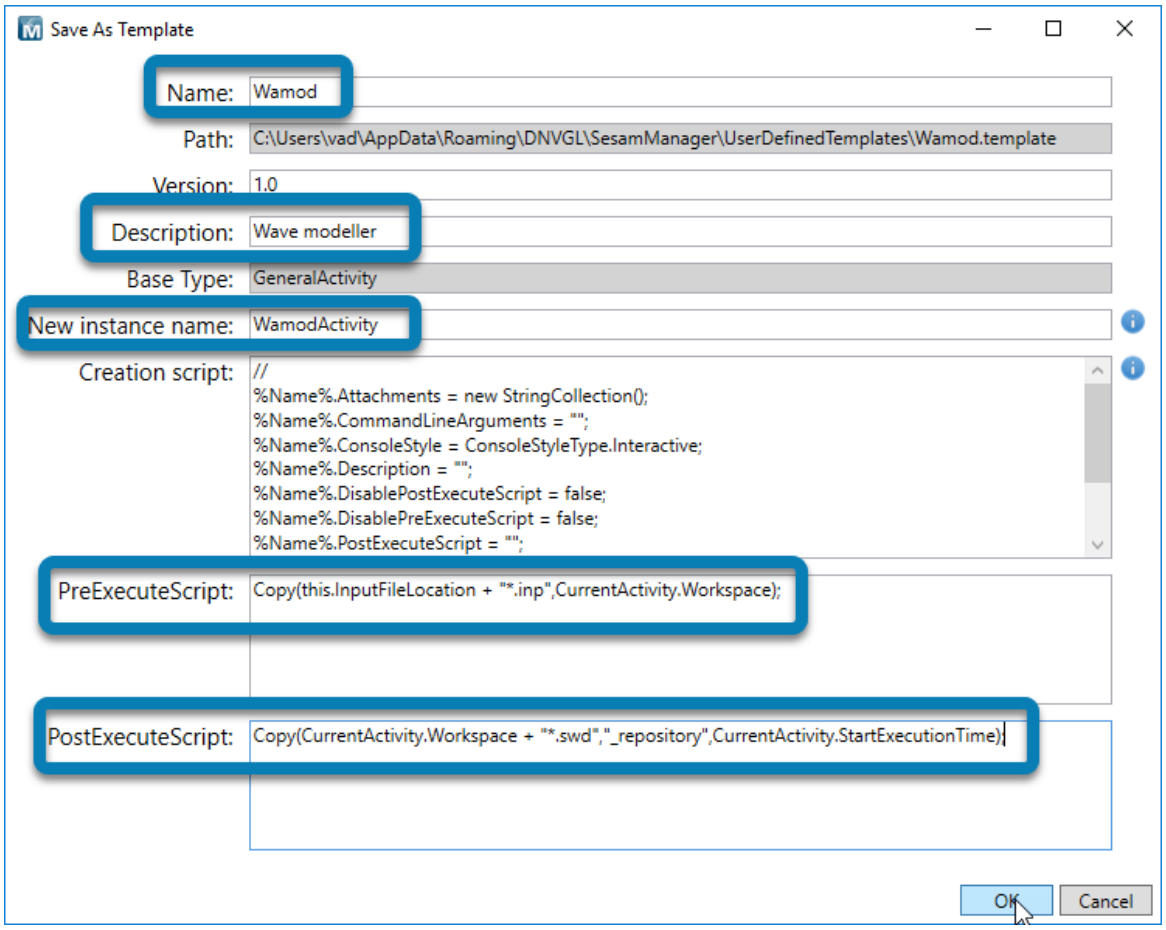


Figure 5.4: Wamod Template

When you press OK, Wamod will appear as a User defined application. You can then create a Wamod activity in your Sesam Manager job by drag and drop. Then specify the name of the .inp file as CommandLineArgument and run the activity. If the inp-file is not in the activity workspace folder it must be specified with either full path or the path relative to the activity workspace folder.

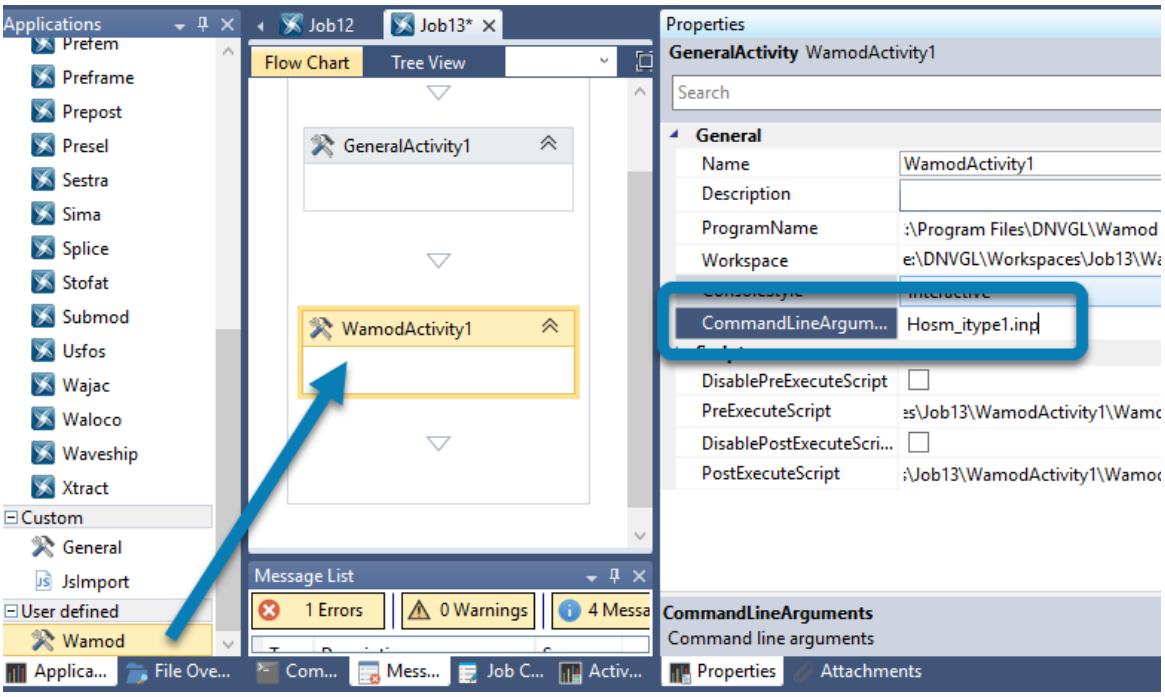


Figure 5.5: Wamod Activity

References

[1] Bitner-Gregersen E. and Gramstad, O. :
"Rogue Waves - Impact on ships and offshore structures"
DNV GL Position Paper 5-2015. DNV GL Strategic Research and Innovation, 2015

[2] Dommermuth, D. G.and Yue, D. K. P.:
"A high-order spectral method for the study of nonlinear gravity waves"
J. Fluid Mech. 184, 267-268, 1987

[3] West, B. J., Brueckner, K. A., Janda, R. S., Milder, D. M.and Milton, R. L.:
"A new numerical method for surface hydrodynamics"
J. Geophys. Res. 92 (C11), 11803–11824., 1987

[4] Zakharov, V. E.:
"Stability of periodic waves of finite amplitude on the surface of a deep fluid"
DNV-RP-205, October 2000

[5] Craig, W. and Sulem, C.:
"Numerical simulation of gravity waves."
J. Comput. Phys. 108 (1), 73–83., 1993

[6] Bateman, W. J. D., Swan, C. and Taylor, P. H.:
"On the efficient numerical simulation of directionally spread surface water waves."
J. Comput. Phys. 174 (1), 277–305., 2001

[7] Choi, J.:
"Notes on formal manipulations of double series."
Korean Math. Soc. 18 (4), 781–789., 2003

[8] Bateman, W. J. D., Swan, C. and Taylor, P. H..:
"On the calculation of the water particle kinematics arising in a directionally spread wave field."
J. Comput. Phys. 186 (1), 70–92., 2003

[9] "Environment Conditions and Environmental Loads, Recommended Practice"
DNV-RP-C205, October 2010

[10] Zhigang Tian, Marc Perlin, and Wooyoung Choi:
"Energy dissipation in two-dimensional unsteady plunging breakers and an eddy viscosity model"
Journal of Fluid Mechanics, 655:217-257, 2010

[11] Zhigang Tian, Marc Perlin, and Wooyoung Choi:
"An eddy viscosity model for two-dimensional breaking waves and its validation with laboratory experiments"
Phys. Fluids, 24(3):036601, 2012

[12] X. Barthelemy, M. L. Banner, W. L. Peirson, F. Fedele, M. Allis, and F. Dias:
"On a unified breaking onset threshold for gravity waves in deep and intermediate depth water"
Journal of Fluid Mechanics, 841:463-488, 2018

[13] Betsy R. Seiffert, Guillaume Ducrozet, and Félicien Bonnefoy:
"Simulation of breaking waves using the high-order spectral method with laboratory experiments: Wave-breaking onset"
Ocean Modelling, 119:94-104, 2017

[14] R. Kurnia and E. van Groesen:
"High order hamiltonian water wave models with wave-breaking mechanism"
Coast. Eng., 93:55-70, 2014.

[15] J. N. Sharma and R. G. Dean:
"Development and evaluation of a procedure for simulating a random directional second order sea surface and associated wave forces"
Technical Report 20, Univ. of Delaware, Newark, 1979

[16] M. M. Rienecker and J. D. Fenton:
"A fourier approximation method for steady water waves"
J. Fluid Mech., 104:119–137, 3 1981

[17] V. P. Krasitskii:
"On reduced equations in the Hamiltonian theory of weakly nonlinear surface-waves"
J. Fluid Mech., 272:1-20, 1994.

About DNV

We are the independent expert in risk management and quality assurance. Driven by our purpose, to safeguard life, property and the environment, we empower our customers and their stakeholders with facts and reliable insights so that critical decisions can be made with confidence. As a trusted voice for many of the world’s most successful organizations, we use our knowledge to advance safety and performance, set industry benchmarks, and inspire and invent solutions to tackle global transformations.

Digital Solutions

DNV is a world-leading provider of digital solutions and software applications with focus on the energy, maritime and healthcare markets. Our solutions are used worldwide to manage risk and performance for wind turbines, electric grids, pipelines, processing plants, offshore structures, ships, and more. Supported by our domain knowledge and Veracity assurance platform, we enable companies to digitize and manage business critical activities in a sustainable, cost-efficient, safe and secure way.

DNV | digital@dnv.com | dnv.com/digital