2nd-order Forces Within HydroDyn

Implimentation

A. Platt

December 4, 2014

Table of Contents

List of Figures

List of Tables

Chapter 1

Overview

This document serves two functions: first, as a guide to how second order forces and bi-directional waves should be incorporated into HydroDyn, and second to document the actual changes as they are made. While Tiago Duarte was visiting NREL over a 6 month period in 2012 and 2013, he sketched out in detail how to incorporate both second order effects and bi-directional waves into HydroDyn. This document is primarily based on his paper for the *AIAA* SciTech 2014 conference [1].

The second order forces can be developed in a standalone sub-module of HydroDyn called WAMIT2 following the FAST framework. This module will calculate the second order forces using information from the Waves sub-module, inputs from the HydroDyn module (including options from the input file), and data files produced by WAMIT. This module will be developed to use bi-directional waves, though testing will be somewhat limited due to the availability of a complete second order solution with bi-directional waves, in the form of a quadratic transfer function (QTF), produced by WAMIT.[1]

The addition of bi-directional waves to HydroDyn will require modification of existing code. Since HydroDyn is still being developed, this addition will take place in the latter stages of the WAMIT2 module development. The bi-directional waves implimentation will use the equal energy approach outlined by Tiago in his document titled "Multi-Directional Waves: Comparison and Implimentation."

[1]Due to the complexity of the calculations, it has been estimated that a full QTF with 57 by 57 frequencies and 37 by 37 wave directions would take on the order of 52 years to calculate with our existing single threaded implimentation of WAMIT.

Chapter 2

Background information

HydroDyn is currently undergoing a conversion to the FAST modularization framework. During this conversion, some additional capabilities are being added to HydroDyn including jacket platforms. At present, HydroDyn does not account for second order wave forces: those forces that arise from the sum and difference of the frequencies of incident waves. HydroDyn also only includes uni-directional waves where all waves are incident from the same direction.

At the start of a FAST simulation, HydroDyn is initialized and calculates the wave spectrum[1] and time history using the *waves* sub-module. This information is then used by the *WAMIT* sub-module and a few other sub-modules to calculate the first order wave foreces on the platform. Because the frequency, direction, and phase of each of the wave components is known at initialization, a time series of the second order forces can be calculated using the *WAMIT2* sub-module before the simulations begins. This requires that the following assumptions are made:

- the wave information in frequency space is known before the start of the simulation (the time series can be found through an FFT),
- the time series of second order forces will act on a single point at the origin (as the first order forces do),
- wave forces and moments are fixed to a reference frame that translates (but does not rotate) with the platform,
- Wavelengths are large compared to the motion of the platform.

2.1 Platform Displacement and Rotation

The complete timeseries for the forces and moments comprising the loads from the first and second order waves are calculated during the initialization of a FAST simulation. These calculated loads are given at the platform origin in the platform rest reference frame and applied during the simulation. During the simulation, the platform will translate and rotate as a result of these loads, aerodynamic loads, and other real time loads (mooring lines etc.). The calculation of the first and second order wave loads do not take into account this displacement.

During the simulation, the FAST glue code calls HydroDyn to calculate the platform loads at each timestep. Since the first- and second-order wave load timeseries were calculated during the initialization, these values are looked up at each timestep and reported back to the glue code on a point mesh (includes translation and rotation).

[1] The wave spectrum is complex-valued and therefore includes all the phase information, so the time history of the wave can be found easily through an FFT.

Chapter 3

Multi-directional Waves

In order to incorporate bi-directional waves, changes were made to the input files, how the data is stored internally, and to the computation of the first order forces. The second order forces were designed from the beginning to include bi-directional waves.

For our implimentation of the multi-directional waves, we are using the equal energy discretization which is used in the commercial code OrcaFlex. With this method the same $N/2+1$ frequencies as used in the uni-directional case are used here. A total of Θ (WaveNDir in input file) discrete directions are used and each frequency is assigned to one of the discrete directions. The value of $(\frac{N}{2})/\Theta$ needs to be an integer so that each direction contains the same number of frequencies. Θ may need to be adjusted within the *Waves* module to ensure this is true (see Section 3.1).

```
  0.0        WaveDir        ! Incident wave propagation mean heading direction                (degrees)
  1          WaveDirMod     ! Directional spreading function {0: unused, 1: COS2S}               (-)
 15          WaveDirSpread  ! Wave direction spreading coefficient ( > 0 )                       (-)
 11          WaveNDir       ! Number of wave directions [odd number only]                        (-)
 90          WaveDirRange   ! Range of wave directions (full range: WaveDir +/- WaveDirRange/2)   (degrees)
```

Table 3.1: *New section for the HydroDyn input file for multi-directional waves. This section is inserted where* WaveDir *is currently defined.*

If WaveDirMod $= 1$, then a check is performed to make sure that WaveMod =2, 3, or 4 (JONSWAP, white-noise, user defined). If this is true, then an internal logical variable WaveMultiDir is set to true and passed to modules that need to know about multiple wave directions.

Within the code in the *HydroDyn_Input* module, the maximum and and minimum directions actually used stored as WaveDirMin and WaveDirMax. Since the wave direction assignments are performed using the equal energy approach, the actual maximum and minimum values will cover a smaller range than requested by the WaveDirRange variable (see Section 3.2.2).

$$\mathsf{WaveDirMin} > \mathsf{WaveDir} - \mathsf{WaveDirRange}/2 \tag{3.1}$$

$$\mathsf{WaveDirMax} < \mathsf{WaveDir} + \mathsf{WaveDirRange}/2. \tag{3.2}$$

These variables will be used in checking the WAMIT output files used (both first and second order). Care should be taken to allow for crossing the $\pm\pi$ direction boundary.

3.1 Adjusting Θ within the *Waves* module

To use the equal energy approach outlined below, $(\frac{N}{2})/\Theta$ must be an integer, so it may be necessary to change Θ from the user specified value. The difficulty with enforcing this arises from three requirements:

1. WaveNDir is odd in order to keep the center direction
2. N is adjusted by HydroDyn to be a product of smallish numbers for efficiency in the FFT
3. $(\frac{N}{2})/\Theta$ needs to be an integer to keep the energy distribution correct[1]

The third requirement means that $N/2$ is a product of integers such that Θ is one of them. For small values of Θ, this is unlikely to be an issue when N is large, but could be problematic for short simulations where N is smaller.[2] For large values Θ, it becomes less likely this would be true. In order to satisfy these conditions, the value of Θ will be adjusted.[3]

For small values of Θ, it may only be necessary to increase it slightly to satisfy the requirements (for example, change $\Theta = 7$ to $\Theta = 9$). For large values of Θ (perhaps > 60), it may be necessary adjust the value several times before a suitable value is found. This could be done as outlined in Figure 3.1. Note that we do not want to simply allow Θ to increase until a suitable value is found in the unlikely event that $N/2$ is a prime number. In this case Θ will have to be increased until $\Theta = N/2$, which would mean each frequency has a unique wave heading and no binning occured. This can only be corrected by having the user change `WaveDT` or `WaveTMax` since N is adjusted internally to satisfy condition 2.

3.2 Equal Energy Approach

Table 3.2: *Notation for multi-directional wave equations.*

Variable	Variable Name	Description	Units
Θ	`WaveNDir`	User defined number of wave directions	-
θ		Current wave direction	degrees
$\bar{\theta}$	`WaveDir`	Central (mean) wave direction	degrees
$\delta\theta$	`WaveDirRange`	Range of wave directions (full width)	degrees
S	`WaveDirSpread`	Spreading function coefficient (1 typ)	-

This method to simulate multi-directional waves assigns each frequency component of the wave to one of Θ discrete wave directions. Each wave direction will have the same number of frequencies, $(\frac{N}{2})/\Theta$, assigned to it. In order to preserve the energy distribution in the wave spreading function, the wave directions are assigned so that a greater number of directions are concentrated near the central frequency.

This method is only valid if the equation describing the total spectrum is separable into the frequency and direction parts as follows:

$$S(\omega,\theta) = \bar{S}(\omega)\cdot D(\theta). \tag{3.3}$$

The wave spreading function is given by

$$D(\theta) = C\left|\cos\left(\frac{\pi\left(\theta-\bar{\theta}\right)}{\delta\theta}\right)\right|^{2S}, \tag{3.4}$$

[1] Originally the requirement was set such that $(\frac{N}{2}-1)/\Theta$ was used (the $\omega = 0$ frequency was ignored). However, it was found that $(\frac{N}{2}-1)/\Theta$ was often a prime number, which would lead to user frustration.

[2] N is initially set to `TMax`/`WaveDT` and then increased until it is the product of small numbers.

[3] While it is theoretically possible to adjust N such that the third condition is satisfied, this would have the undesirable consequence of changing the frequency step (see Equation (6.2)). This would in turn change the values of the complex wave amplitude in frequency space (A_m) for a given random number seed value. Subsequently, the ability to reproduce the wave height and wave force timeseries for a given seed pair would not be preserved.

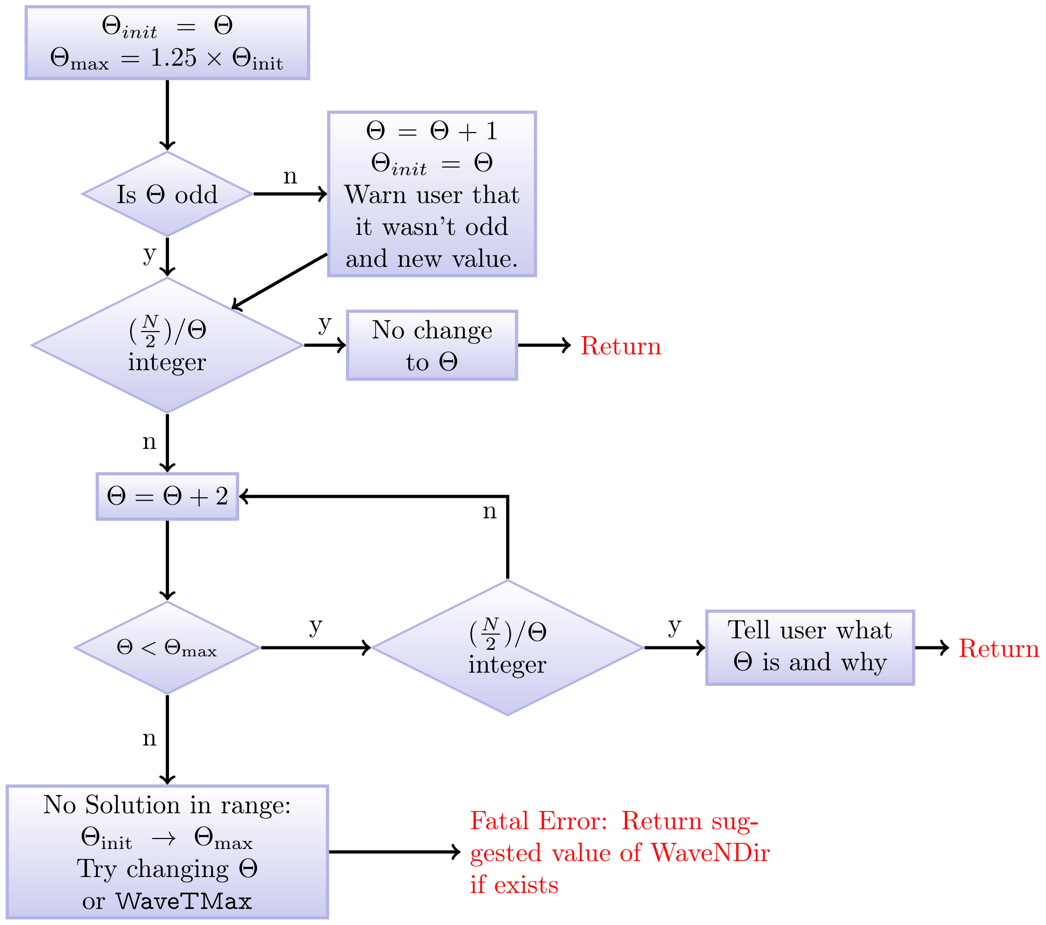
$\Theta_{init} = \Theta$
$\Theta_{\max} = 1.25 \times \Theta_{\text{init}}$
Is Θ odd
n
$\Theta = \Theta + 1$
$\Theta_{init} = \Theta$
Warn user that it wasn't odd and new value.
y
$(\frac{N}{2})/\Theta$ integer
y
No change to Θ
Return
n
$\Theta = \Theta + 2$
n
$\Theta < \Theta_{\max}$
y
$(\frac{N}{2})/\Theta$ integer
y
Tell user what Θ is and why
Return
n
No Solution in range: $\Theta_{\text{init}} \rightarrow \Theta_{\max}$ Try changing Θ or WaveTMax
Fatal Error: Return suggested value of WaveNDir if exists

Figure 3.1: *Flowchart of how to find a suitable value of Θ such that $(\frac{N}{2})/\Theta$ is an integer. The reason for suggesting that* WaveTMax *should be changed is that there is a possibility that $N/2$ is a prime number. This is implimented at the start of the initialization subroutine of the* Waves *module if multidirectional waves are selected.*

where the normalization constant, C, is given by

$$C = \frac{\sqrt{\pi}\,\Gamma(S+1)}{\delta\theta\,\Gamma(S+1/2)}, \tag{3.5}$$

and Γ is the gamma function. The spreading function should satisfy the normalization condition of

$$\int\limits_{\bar{\theta}-\delta\theta/2}^{\bar{\theta}+\delta\theta/2} D(\theta)\,\mathrm{d}\theta \equiv 1. \tag{3.6}$$

The cumulative energy distribution within the wave spreading function up to angle θ is given by

$$P(\theta) = \int\limits_{\bar{\theta}-\delta\theta/2}^{\theta} D(\theta')\,\mathrm{d}\theta'. \tag{3.7}$$

The following method may be used to set the appropriate wave directions to satisfy the equal energy approach.

1. Discretize the wave direction range $\delta\theta$ by n_d steps and calculate $D(\theta)$ spreading function. Set n_d to a sufficiently large number such that the function is smooth enough for interpolation over Θ directions (set $n_d = 3\Theta$).
2. While calculating $D(\theta)$, calculate the cumulative energy sum up to the current direction as $P(\theta)$.
3. Discretize $P(\theta)$ into Θ steps from $1/\Theta \leq P_i \leq 1 - 1/\Theta$. Interpolate the function $D(\theta)$ found in step 1 to find the corresponding values of θ_i. The θ_i values are the wave directions used in the equal energy method.
4. Randomly assign each of the $N/2$ frequencies (ignoring the end frequencies at which the wave amplitude is defined as zero) to a θ_i direction such that each wave direction has $(N/2)/\Theta$ frequencies assigned to it. *This is the tricky bit to do such that the assignments are the same given a certain random number seeed pair.*

3.2.1 A few notes

There are a few things to take note of in this process. First, in order to preserve the equal energy distribution within the wave direction spreading function, care must be taken so that $(N/2)/\Theta$ is an integer (see Section 3.1 for caveats with using a user defined value for Θ). Second, if it is desirable to have a greater number wave directions, it is preferable to increase Θ substantially. Alternatively, the θ_i values could be treated as wave direction bins spanning a range of θ directions rather than a single direction. The danger with this is that the distribution of random wave directions within the bin will affect the overall energy distribution of the spreading function. A shaping function would then need to be applied to the bin to distribute wave directions within it. In this case it would be preferable to increase Θ significantly in order to allow for more wave directions while preserving the energy distribution in the wave spreading function. Short tests using MATLAB code indicate that a value of $\Theta = 399$ works well for allowing a seemingly random set of wave directions. In practice, it may be preferable to keep $\Theta < 50$ for a more physical description of wave directionality.

3.2.2 Implimentation and Testing

Development testing During the implimentation of the multidirectional waves within the *Waves* module, testing was performed to ensure that no errors were introduced into the equations used to calculate the wave elevation, velocity, and acceleration. The following sequence was used to impliment the multidirectional waves within the waves module.

1. Add code section to change the value of `WaveNDir`. This was originally implimented such that $(\frac{N}{2}-1)/\Theta$ was an integer. This was later changed when it was discovered that $(\frac{N}{2}-1)$ is often a prime number (due in part to how `NStepWave` is adjusted – see Section 3.1 for details). This code was tested before proceeding.
2. Split the main loop over `NStepWave2` frequencies. The wave direction assignment loop is inserted between the first loop where the wave amplitudes and phases are assigned, and the second loop where the IFFTs are setup and performed. Testing was performed to ensure that all the information necessary in the second loop existed.
3. Create the wave direction assignment loop. Some temporary print statements were used to output the wave direction information for plotting. These results are shown in Figures 3.2 to 3.7.
4. The code necessary for processing the input array `WaveElevXY` and return array `WaveElevSeries` was developed at the *Waves* module and the *WAMIT2* driver level. The subroutine for calculating the wave elevation at a specified (x, y) coordinate was developed. Movies were then created of the test cases listed in Table 3.3 to verify that the wave elevation calculations were working correctly (it too some iterating to get this working).
5. The code was then rearranged and modified so that the `WaveElevXi`, `WaveElevYi` elevation data uses the wave elevation routine.
6. Now the wave velocity and acceleration equations were modified. In the original calculations, the wave direction was handled after the FFT. This was revised so that the wave direction was handled before the FFT was calculated. Testing of this was performed by setting `WaveNDir` $=$ 1 and repeating a few CertTests that had been generated with the previous version of the *Waves* module. The results agreed to within the precision used for outputting the CertTest results.

At this point, it was concluded that the *Waves* module was working as well as it had been prior to the implimentation of multi-directional waves. A minor usability issue became apparent at this point regarding how `WaveNDir` was adjusted within the code. The decision was then made to modify the code such that we include the $\omega = 0$ term in the wave direction assignment. This means that we now force $(\frac{N}{2})/\Theta$ to be an integer. We also included code to give the user some idea of what values of `WaveNDir` might work with their currently defined values of `WaveTMax` and `WaveDT`.

For testing the multi-directional waves there is one more test we can perform. This is to check the correlation of the wave elevation at various (x, y) locations. This has not been performed yet.

do this

Test Cases Multidirectional waves using the method described above are implimented within the *Waves* module of HydroDyn. It is evaluated after the generation of the wave spectrum (JONSWAP or other). The directions are assigned in groups of Θ frequencies so that one each frequency within the group is assigned a unique direction. This is repeated for all $(N/2)/\Theta$ groups of frequencies.

Several test cases were performed to verify that the assignment of wave directions by frequency was correctly performed. In each of the test cases listed in Table 3.3, a compiled version of the code with partially implimented modifications to the *Waves* module was executed. The calculated spreading function, integrated power of the spreading function, calculated wave directions, and the assigned wave directions were output using appropriately placed print statements (this was merely intended as an intermediate testing of the code, not a test that would be needed in this form later). The results were plotted using gnuplot.

Tests numbered 002 and 004 are very similar with the only differences in $\bar{\theta}$, the mean wave direction, and S, the spreading coefficient. A comparison of Figure 3.3 and Figure 3.5 shows the repeatablility during the assignment of the wave direction for each frequency: in each test, the n^{th} frequency is assigned to the same wave direction index. Though the frequencies are randomly assigned, the same seed was used in each of these tests and the random number generator had the same number of calls for wave amplitude prior to the assignment of the directions.

Tests numbered 001, 005, and 008 are identical with the exception of the mean wave direction.

Table 3.3: *Multi-directional waves test case parameters.*

Test Case (-)	WaveDirMod	$\bar{\theta}$ (deg)	$\delta\theta$ (deg)	S (-)	$\tau_{\max}$ (s)	$\Delta\tau$ (s)	$N/2$ (-)	Θ (-)	$\frac{N/2}{\Theta}$ (-)
001	1	0	50	1.0	85	0.25	170	17	10
002	1	45	60	1.0	150	0.25	300	25	12
003	1	-137	45	1.0	390	0.25	780	39	20
004	1	135	45	2.3	150	0.25	300	25	12
005	1	90	50	1.0	85	0.25	170	17	10
006	0	45	60	1.0	150	0.25	300	–	–
007	0	-137	45	1.0	390	0.25	780	–	–
008	1	15	35	1.0	85	0.25	170	17	10

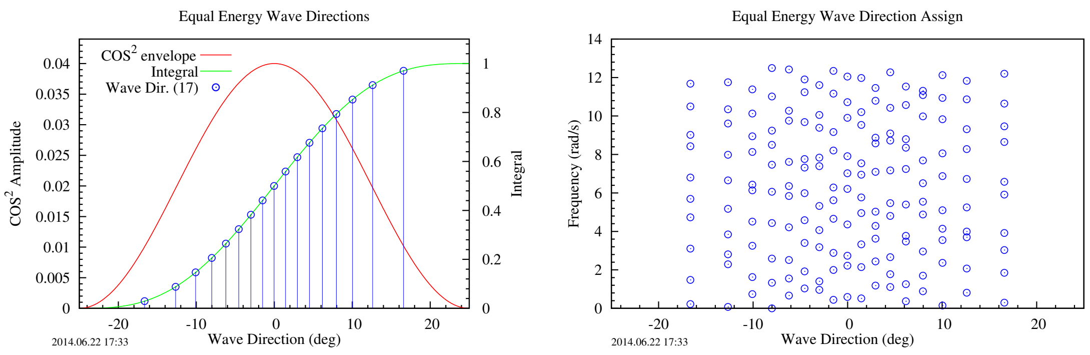

Figure 3.2: *Test case 001. The right plot shows the randomly selected directions for each frequency.*

Test 001 is oriented with the mean wave direction along the x axis, and test 005 is oriented with the mean wave direction along the positive y axis. A movie of the sea surface for these two tests is useful in checking that the coordinate transformations are working correctly.

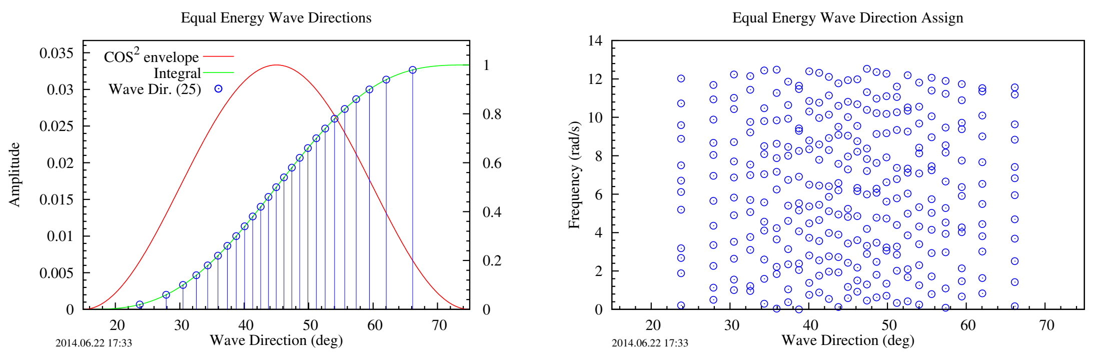

Figure 3.3: *Test case 002. The right plot shows the randomly selected directions for each frequency.*

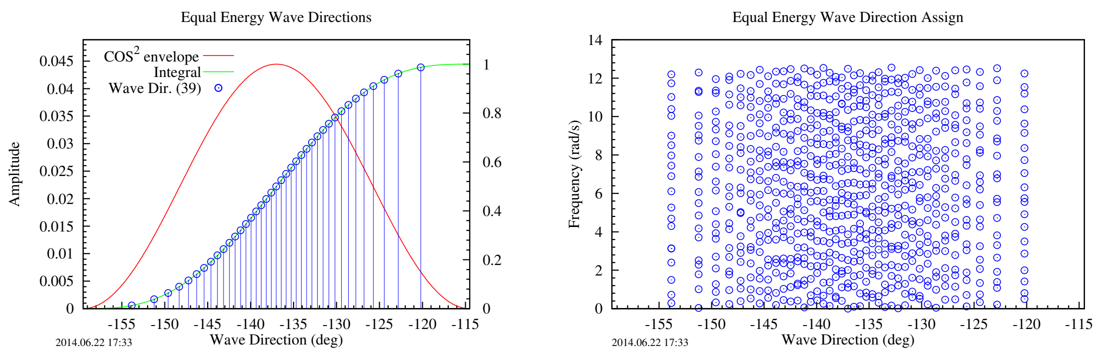

Figure 3.4: *Test case 003. The right plot shows the randomly selected directions for each frequency.*

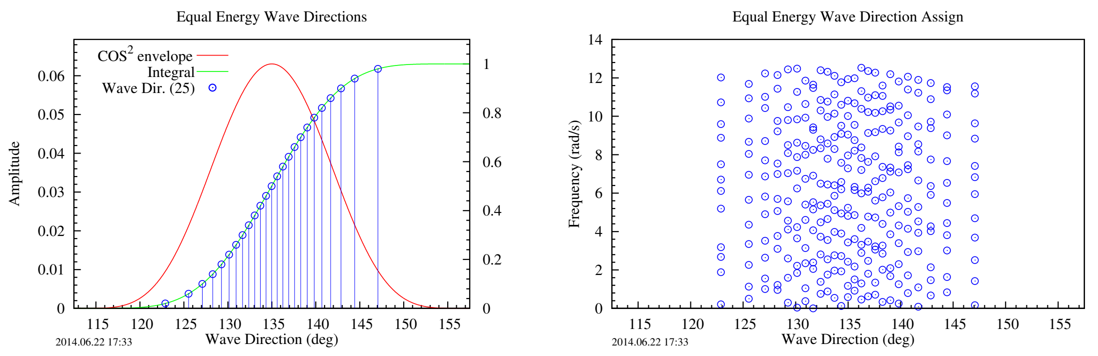

Figure 3.5: *Test case 004. The right plot shows the randomly selected directions for each frequency. This is the same as test case 002 with $\bar{\theta} = 135$ and $S = 2.3$. Note that the ordering of the assigned frequencies is the same as in case 002.*

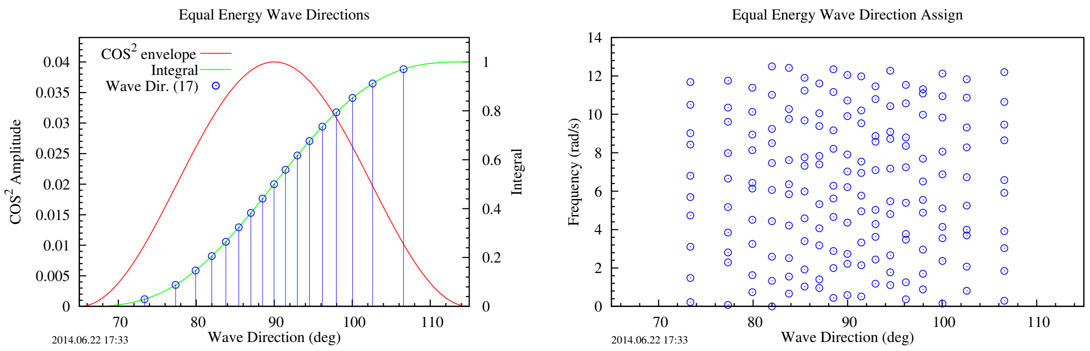

Figure 3.6: *Test case 005. The right plot shows the randomly selected directions for each frequency.*

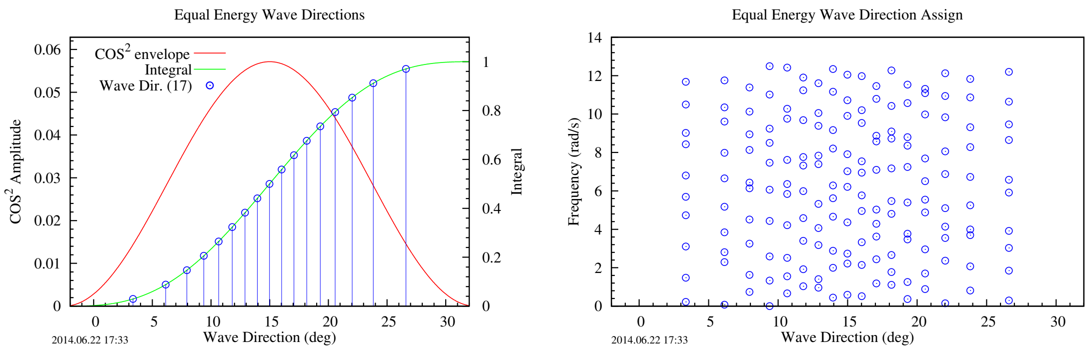

Figure 3.7: *Test case 008. The right plot shows the randomly selected directions for each frequency.*

3.3 Changes during implimentation

Waves module: `InitOutputType%WaveDir` will continue to contain the mean wave heading. A new array `InitOutputType%WaveDirArr` will contain the direction headings for each wave elevation (same number of elements as `WaveElevCO`).

Waves module: For calculations of wave height, velocity, and acceleration away from the origin, modifications were made to split the wave into x and y components and calculate each separately (used to be single component along wave direction).

Waves module: the equations for the wave velocity and acceleration used within the *Waves* module are modified to accomodate the wavedirections as a function of frequency. In Jason's dissertation, the $cos(\beta)$ and $sin(\beta)$ terms in equation 2-31a, 2-31b, 2-32a, and 2-32b are moved inside the IFFT.

WAMIT module: Change wording about `WaveDir` to correspond to the mean wave direction. Add in a new variable `WaveDirArr` to handle the directions for each frequency.

WAMIT module: Change the code to use `WaveDirArr` at each frequency. This involved combining what had been two one-dimensional interpolations into a single two-dimensional interpolation scheme.

In addition to the above changes, a new pair of input and output arrays were specified to allow for the calculation of the wave elevation at arbitrarily specified (X,Y) coordinates. The input array, `WaveElevXY`, of size $2 \times N$ allows the for a set of N arbitrary number of (x, y) coordinates to be specified. For this array, index 1 corresponds to specifies which coordinate (x or y) and index 2 corresponds to point number. If `WaveElevXY` has been allocated at the gluecode or driver level, an array, `WaveElevSeries`, of size `NStepWave`$\times N$ is returned. The first index of this array is the timestep (of the `NStepWave` timesteps in the simulation), and the second index corresponds to the point number (of N points) specified in the `WaveElevXY` input array. This has been implimented both within the *Waves* module and within HydroDyn itself. It has been tested with the HydroDyn driver and with the *WAMIT2* driver code to generate sea surface movies corresponding to each of the 8 tests listed in Table 3.3.

Chapter 4

WAMIT: Output Files

Table 4.1: *Notation for data contained in the WAMIT output files. Here $\bar{F}$ is the force quadratic transfer function. The units listed are those given within the WAMIT manual [2]. Note: some variable names appear differently here than in the WAMIT manual.*

Variable	Type	Units
τ	Wave period	Seconds
β	Wave direction	Degrees
k	Force component	None
$\lvert\bar{F}\rvert$	Force QTF Magnitude	Nondimensional
θ	Force QTF Phase	Degrees
$\Re(\bar{F})$	Force QTF Real part	Nondimensional
$\Im(\bar{F})$	Force QTF Imaginary part	Nondimensional

WAMIT can use several methods to calculate the normalized second order wave force. Each of those methods produces a different type of output file. The calculation methods available to the WAMIT2 module are limited by the WAMIT output files that are available. Table 5.1 shows what second order force calculation methods are available depending on the WAMIT files available [2]. Table 4.1 lists the variables used in this document to refer to the various WAMIT outputs. Note that the notation here differs with the notation used in the WAMIT manual and in Tiago's writings [1, 2].

4.1 WAMIT Output

Table 4.2 lists the format for each of the WAMIT output files. Within these files, the angles and frequencies do not need to be evenly spaced. Also, it should also be noted that the discretization between ω_1 and ω_2 are not necessarily the same. This also applies to the angles β_1 and β_2.

A value for the period $\tau = 0$ in the WAMIT output file means that the frequency $\omega = \infty$. A period of $\tau < 0$ is used in the WAMIT output file to indicate $\omega = 0$.

The QTF in the WAMIT output files does not need to be complete since the following relationships are true:

$$\bar{F}^{-}_{mn} = \bar{F}^{-\,*}_{nm} \qquad \text{and} \qquad \bar{F}^{+}_{mn} = \bar{F}^{+}_{nm}, \tag{4.1}$$

where $*$ indicates the complex conjugate. Note that this implies that the diagonal terms in the second order difference QTF, $\bar{F}^{-}(\omega_m, \omega_m) = \bar{F}^{-\,*}(\omega_m, \omega_m)$, are real valued.

Table 4.2: *WAMIT output files and format (version 6.4/6.1s) [2]. In this table, τ is the period of the wave, β_m and β_n are the wave directions. The second order complex force term is given by both the amplitude and phase pair ($\left|\bar{F}_m\right|$ and θ_m), and in terms of its real and imaginary parts ($\Re(\bar{F}_m)$ and $\Im(\bar{F}_m)$). The index k indicates the force ($k = 1 \ldots 3$) or moment ($k = 4 \ldots 6$) load components. The indices m and n correspond to the period of the waves that are interacting (see pages 4-9 and 11-13 of Ref. 2).*

File Ext.	Description	Output Format								
.7	Mean drift based on momentum flux (WAMIT v. 7 only)	τ_m		$\beta_{m(1)}$	$\beta_{m(2)}$	k	$\left\lvert\bar{F}^-_{mm}\right\rvert$	θ_{mm}	$\Re(\bar{F}^-_{mm})$	$\Im(\bar{F}^-_{mm})$
.8	Mean drift (modes 1, 2, and 6)	τ_m		$\beta_{m(1)}$	$\beta_{m(2)}$	k	$\left\lvert\bar{F}^-_{mm}\right\rvert$	θ_{mm}	$\Re(\bar{F}^-_{mm})$	$\Im(\bar{F}^-_{mm})$
.9	Mean drift (all modes)	τ_m		$\beta_{m(1)}$	$\beta_{m(2)}$	k	$\left\lvert\bar{F}^-_{mm}\right\rvert$	θ_{mm}	$\Re(\bar{F}^-_{mm})$	$\Im(\bar{F}^-_{mm})$
.10s	Quadratic 2nd-order sum forces	τ_m	τ_n	β_m	β_n	k	$\left\lvert\bar{F}^+_{mn}\right\rvert$	θ^+_{mn}	$\Re(\bar{F}^+_{mn})$	$\Im(\bar{F}^+_{mn})$
.10d	Quadratic 2nd-order difference forces	τ_m	τ_n	β_m	β_n	k	$\left\lvert\bar{F}^-_{mn}\right\rvert$	θ^-_{mn}	$\Re(\bar{F}^-_{mn})$	$\Im(\bar{F}^-_{mn})$
.11s	Total 2nd-order forces by indirect method (sum)	τ_m	τ_n	β_m	β_n	k	$\left\lvert\bar{F}^+_{mn}\right\rvert$	θ^+_{mn}	$\Re(\bar{F}^+_{mn})$	$\Im(\bar{F}^+_{mn})$
.11d	Total 2nd-order forces by indirect method (diff)	τ_m	τ_n	β_m	β_n	k	$\left\lvert\bar{F}^-_{mn}\right\rvert$	θ^-_{mn}	$\Re(\bar{F}^-_{mn})$	$\Im(\bar{F}^-_{mn})$
.12s	Total 2nd-order forces by direct method (sum)	τ_m	τ_n	β_m	β_n	k	$\left\lvert\bar{F}^+_{mn}\right\rvert$	θ^+_{mn}	$\Re(\bar{F}^+_{mn})$	$\Im(\bar{F}^+_{mn})$
.12d	Total 2nd-order forces by direct method (diff)	τ_m	τ_n	β_m	β_n	k	$\left\lvert\bar{F}^-_{mn}\right\rvert$	θ^-_{mn}	$\Re(\bar{F}^-_{mn})$	$\Im(\bar{F}^-_{mn})$

The normalized second order wave force in the WAMIT output files is non-dimensional (see Ref [2] section 11.6 for details). These are written as

$$\bar{F}^-_k = \frac{\tilde{F}^-_k}{\rho g L^a A_m A_n^*} \qquad \text{and} \qquad \bar{F}^+_k = \frac{\tilde{F}^+_k}{\rho g L^a A_m A_n}, \tag{4.2}$$

where $a = 1$ for forces ($k = 1, 2, 3$) and $a = 2$ for moments ($k = 4, 5, 6$), $\bar{F}_k$ and $\tilde{F}_k$ are respectively the non-dimensioned and dimensioned QTFs for the k^{th} force component, and A_m and A_n^* are the complex wave and complex conjugates of the amplitudes. Other variables are listed in Table 5.2. In order to simplify using the dimensioned wave force in calculations, Equation (4.2) can be rewritten as

$$\tilde{F}^-_k = A_m A_n^* \left(\bar{F}^-_k \rho g L^a\right) = A_m A_n^* F^-_k \qquad \text{and} \qquad \tilde{F}^+_k = A_m A_n \left(\bar{F}^+_k \rho g L^a\right) = A_m A_n F^+_k, \tag{4.3}$$

where $F^\pm_k$ is partially dimensioned by $\rho g L^a$, but does not include the wave amplitudes.

4.2 Reading WAMIT Data Files

Make pretty version of Figure 4.1

4.2.1 First order WAMIT output files (.3)

The algorithm used in reading in the first order WAMIT data files involves scanning through the data file multiple times. This is roughly summarized as:

1. Get wave period information
   (a) Read through file to find the number of wave periods

   (b) Allocate array to hold the wave periods in the order they appear in the file (`WAMITPer` and `WAMITFreq`)
   (c) Allocate array `SortFreqInd` to hold the ordered indices of the sorted frequencies
   (d) Allocate array `HdroFreq` to hold the sorted frequencies
   (e) Read through file to populate the `WAMITPer`, `WAMITFreq` and `SortFreqInd` arrays
   (f) Read through file and store the sorted frequencies in `HdroFreq`

2. Get wave directions
   (a) Read through first wave period to get the directions
   (b) Allocate array to hold the wave directions in the order they appear in the file (`WAMITDir`)
   (c) Allocate array `SortWvDirInd` to hold the ordered indices of the sorted directions
   (d) Allocate array to hold the ordered indices of the sorted directions (`HdroWvDir`)
   (e) Read through file to populate `WAMITDir` and assemble
   (f) Read through file and store the sorted directions in `HdroWvDir`

3. Populate the complex wave information `HdroExctn`
   (a) Read through the file again to populate `HdroExctn`(`SortFreqInd`(K),`SortWvDirInd`(J),I)

There are two subtle assumptions about the organization of the WAMIT output file that are made here. First, the file is always organized such that the wave direction is looped through for each value of the wave period. The second assumption is that all the force components are grouped for a given wave direction and wave period. In the case of the second order WAMIT data, this may not be true.

4.2.2 Second order WAMIT output files

The calculations for the second order WAMIT output are very time consuming. In order to help alleviate this problem, WAMIT allows the user to specify exactly which combinations of wave heading and wave periods to calculate in a .PT2 input file. As a result, the output file may be very sparse, and may not be ordered in any meaningful way. Therefore, a different approach that does not make assumptions about the ordering of the file should be used.

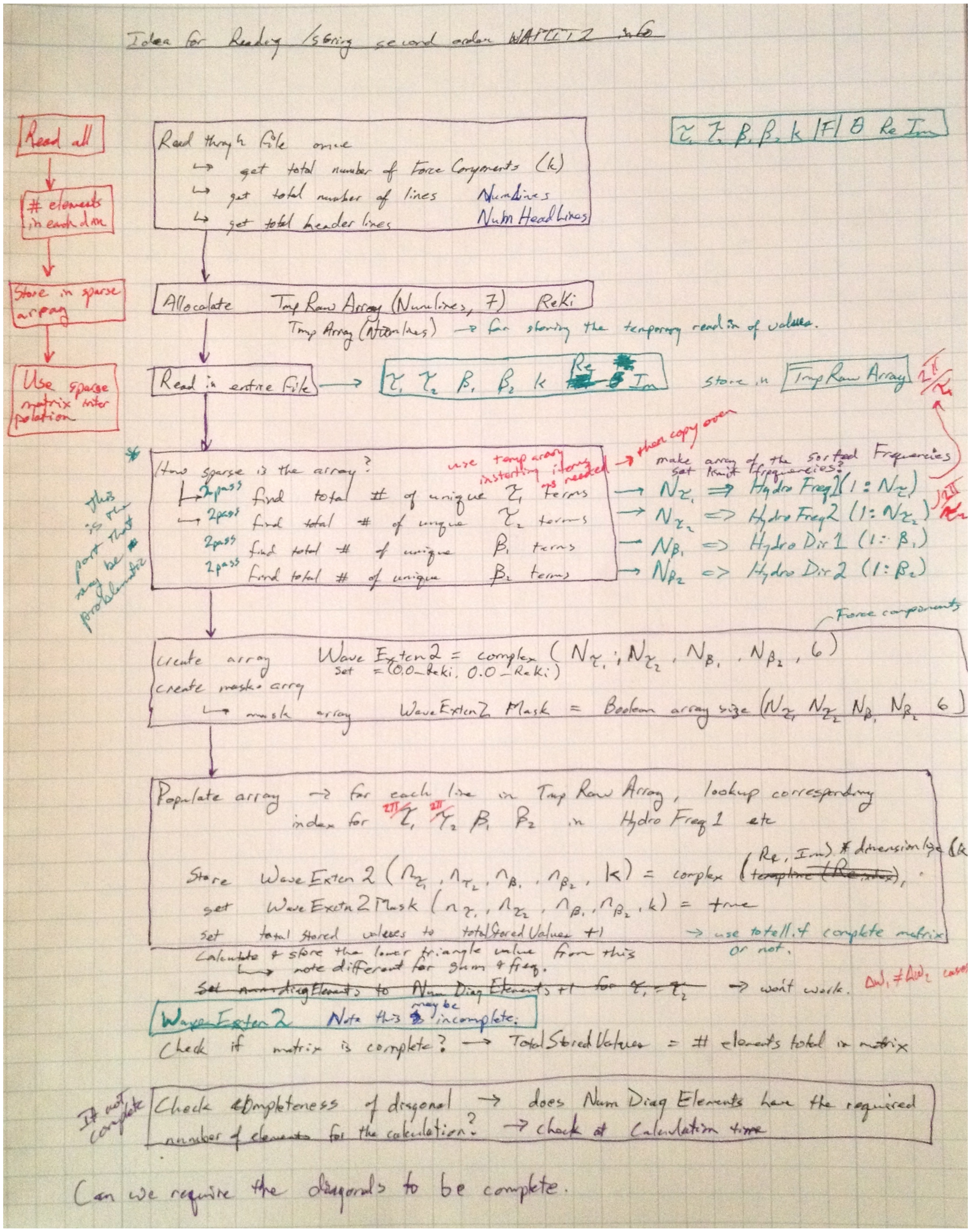
Idea for Reading / Storing second order WAMIT2 info
Read all
# elements in each dim
Store in sparse array
Use sparse matrix interpolation
Read through file once
↳ get total number of Force Components (k)
↳ get total number of lines NumLines
↳ get total header lines NumHeadLines
τ₁ τ₂ β₁ β₂ k |F| θ Re Im
Allocate TmpRowArray (NumLines, 7) ReKi
TmpArray (NumLines) → for storing the temporary read in of values.
Read in entire file → τ₁ τ₂ β₁ β₂ k Re Im
store in TmpRowArray
How sparse is the array?
use temp array inserting items as needed → then copy over
2pass find total # of unique τ₁ terms → Nτ₁ ⇒ HydroFreq1(1:Nτ₁)
2pass find total # of unique τ₂ terms → Nτ₂ ⇒ HydroFreq2(1:Nτ₂)
2pass find total # of unique β₁ terms → Nβ₁ ⇒ HydroDir1(1:β₁)
2pass find total # of unique β₂ terms → Nβ₂ ⇒ HydroDir2(1:β₂)
make array of the sorted Frequencies
this is the part that may be problematic
Force components
create array WaveExtn2 = complex (Nτ₁, Nτ₂, Nβ₁, Nβ₂, 6)
set = (0.0_ReKi, 0.0_ReKi)
create mask array
↳ mask array WaveExtn2Mask = Boolean array size (Nτ₁, Nτ₂, Nβ₁, Nβ₂, 6)
Populate array → for each line in TmpRowArray, lookup corresponding index for τ₁ τ₂ β₁ β₂ in HydroFreq1 etc
Store WaveExtn2(nτ₁, nτ₂, nβ₁, nβ₂, k) = complex (Re, Im) * dimensionalize
set WaveExtn2Mask(nτ₁, nτ₂, nβ₁, nβ₂, k) = true
set total stored values to TotalStoredValues +1
→ use to tell if complete matrix or not.
Calculate + store the lower triangle value from this
↳ note different for sum + freq.
~~Set NumDiagElements to NumDiagElements +1 for τ₁ = τ₂~~ → won't work. Δω₁ ≠ Δω₂ cases
WaveExtn2 Note this may be incomplete.
Check if matrix is complete? → TotalStoredValues = # elements total in matrix
If not complete
Check completeness of diagonal → does NumDiagElements have the required number of elements for the calculation? → check at calculation time
Can we require the diagonals to be complete.

Figure 4.1: *Overview of scheme for reading in 2nd order WAMIT files with two periods (.10, .11, .12). The algorithm for reading in 2nd order WAMIT files with only one period (.7, .8, .9) should be a simplified version of this algorithm.*

The overall idea of this approach is to read the entire data file a temporary array in memory, and then figure out how many unique values of $\omega_1, \omega_2, \beta_1$, and β_2 there are. Once this has been determined, the arrays storing the sorted frequencies and directions can be allocated along with the array for storing $F^{\pm}$. In addition to the array holding the data, a mask array (boolean?) of equal size must be created to store information regarding which values are populated. This will be necessary for knowing how sparse the array is and for interpolation algorithms that can handle limited sparseness.

Due to the number of times the file would need to be read for this algorithm, it will likely be faster to read everything into memory and then process rather than rewind the file many times (disk IO is slow). This does impose some memory usage that could be avoided, but given how long it takes WAMIT to perform these calculations, the files are likely to be small (at least in comparison to the wind files).

4.3 WAMIT Data Integrity Checks

A few checks can be enforced to ensure that the calculations within the *WAMIT2* module can be performed using the provided data. If the limits of frequency or wave direction are set outside what is provided within the WAMIT output data, an error should be issued and the program aborted. This could be checked during the calculations by checking if either $\omega < \min(\tilde{\omega})$ or $\omega > \max(\tilde{\omega})$ is true, where $\tilde{\omega}$ is the array of frequencies given in the WAMIT output files. Alternatively, this could be checked immediately after reading the WAMIT output file by checking against the limits in the HydroDyn input file as follows:

$$\texttt{WvLowCOffD} > \min(\tilde{\omega}_D) \tag{4.4}$$

$$\texttt{WvHiCOffD} < \max(\tilde{\omega}_D) \tag{4.5}$$

for difference QTF files, and

$$\texttt{WvLowCOffS} > \min(\tilde{\omega}_S) \tag{4.6}$$

$$\texttt{WvHiCOffS} < \max(\tilde{\omega}_S) \tag{4.7}$$

for sum QTF files where $\tilde{\omega}_D$ and $\tilde{\omega}_S$ are the arrays of frequencies found in the difference and sum WAMIT files, respectively.

In addition to checking the frequency range of the WAMIT output files, the wave direction range should also be checked. The values can be checked agains the `WaveDirMin` and `WaveDirMax` variables (derived from `WaveDir` and `WaveDirRange` in the HydroDyn input file – see Chapter 3). Care needs to be taken to account for a possible boundary between positive and negative direction headings (π and $-\pi$ directions).

Chapter 5

Second Order Force Calculations

In this chapter, the equations and algorithms used to find the second order hydrodynamic forces from the output of WAMIT are discussed. The forms of the equations presented here are what is used in the WAMIT2 module within HydroDyn. Along with the equations, the methodology and processing necessary to take the WAMIT output file types and parse them for use in the equations is also covered in detail. The interpolation algorithms used are covered in Chapter 6.

Table 5.1 shows which calculations methods can be used depending on what WAMIT output files are available. The criteria for downselecting information in the file for use in the calculations is given for each method and file type combination.

The notation used in this chapter is given in Table 5.2 (note that this differs slightly from the notation in the WAMIT manual [2] and in Tiago's writings [1]).

Table 5.1: *Matrix of possible calculation methods and data file combinations. The names given in the column marked* Variables *are the independent variables within the file. The information used from each file type is given under each method. Note that for the mean-drift calculation with multi-directional waves, only the $\beta_1 = \beta_2$ terms are necessary with the equal-energy approach. However, if a different multidirectional waves method is used, then all the data would be necessary.*

Sea	File	Independent	Method			
Directionality	Ext.	Variables	Mean Drift	Newman's	Diff-QTF	Sum-QTF
Uni-Directional Seas	7 8 9	ω β_1, β_2	$\beta_1 = \beta_2$	$\beta_1 = \beta_2$	Not possible	Not possible
	10 11 12	ω_1, ω_2 β_1, β_2	$\omega_1 = \omega_2$ $\beta_1 = \beta_2$	$\omega_1 = \omega_2$ $\beta_1 = \beta_2$	$\beta_1 = \beta_2$	$\beta_1 = \beta_2$
Multi-Directional Seas	7 8 9	ω β_1, β_2	All Data	Not possible	Not possible	Not possible
	10 11 12	ω_1, ω_2 β_1, β_2	$\omega_1 = \omega_2$		All Data	All Data

Table 5.2: *Notation used in the second order force equations. The variable name used in the fortran code is also listed where appropriate.*

Variable	Fortran Variable	Description
A_m		Complex amplitude of the m^{th} frequency given in $Z[m]$ array from the *Waves* module. This includes a $N/2$ term in it.
a_m		Complex amplitude of the m^{th} frequency given in $Z[m]$ array from the *Waves* module. This has the $N/2$ term removed.
i	`ImagNmbr`	The imaginary number $i = \sqrt{-1}$ (i and j are not used as indices here to avoid confusion)
k		Index to load component (translation: $1-3$; rotation: $4-6$)
m		Index to first wave
n		Index to second wave
μ^{-}		Difference index of two waves frequencies ($= m - n$)
μ^{+}		Summation index of two wave frequencies ($= m + n$)
τ	`WAMITPer`	Period of wave in WAMIT file
τ_1	`WAMITPer1`	Period of first wave in pair read from WAMIT file
τ_2	`WAMITPer2`	Period of second wave in pair read from WAMIT file
ω	`Omega`	Frequency (rad/s)
N	`NStepWave`	Total number of timesteps (range 0:N)
$N/2$	`NStepWave2`	Total number of positive wave frequencies (range 0:N/2)
$Z(\omega_m)$	`WaveElevC0`	The discretized complex wave form in the frequency domain
$\Phi(\omega_m)$	`WaveDir`	Wave direction for each wave frequency
$t_{\max}$	`WaveTMax`	The maximum time of the wave simulation
Δt	`WaveDT`	The timestep for the wave simulation
ω_{lo}	`WvLowCOff`	The low frequency cutoff for first order waves
ω_{hi}	`WvHiCOff`	The high frequency cutoff for first order waves
$\Delta\omega$	`WaveDOmega`	The frequency stepsize
$\omega_{\text{lo-d}}$	`WvLowCOffD`	Difference frequency low-cutoff
$\omega_{\text{hi-d}}$	`WvHiCOffD`	Difference frequency high-cutoff
$\omega_{\text{lo-s}}$	`WvLowCOffS`	Sum frequency low-cutoff
$\omega_{\text{hi-s}}$	`WvHiCOffS`	Sum frequency high-cutoff
$\bar{F}_k^{-}$		k^{th} component of the non-dimensional difference frequency transfer matrix (data from WAMIT)
F_k^{-}		k^{th} component of the dimensionalized difference frequency transfer matrix
$\bar{F}_k^{+}$		k^{th} component of the non-dimensional sum frequency transfer matrix (data from WAMIT)
F_k^{+}		k^{th} component of the dimensionalized sum frequency transfer matrix
ρg	`RhoXg`	Water density * gravity
L	`WAMITULEN`	WAMIT characteristic body length scale

5.1 Overview

In order to simplify the data processing as much as possible, the WAMIT output data will sorted and stored in either a three-dimensional matrix for first order derived transfer matrices (.7, .8, and .9 files), or a four-dimensional for data derived by full QTF methods (.10, .11, and .12 files). The values needed for each step in the calculation are interpolated from these arrays using either three-

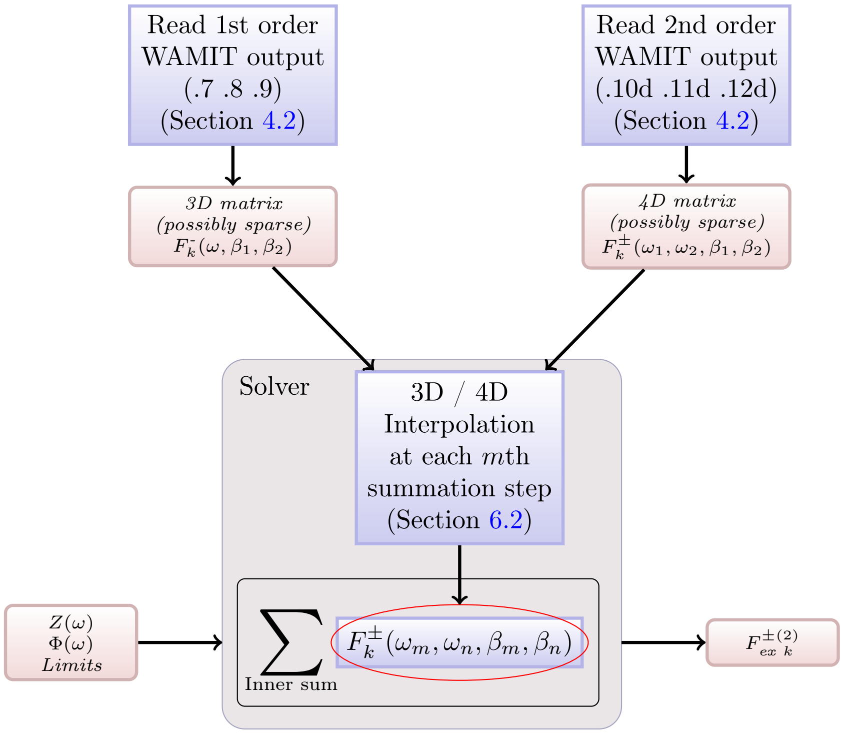
Read 1st order WAMIT output (.7 .8 .9) (Section 4.2)
Read 2nd order WAMIT output (.10d .11d .12d) (Section 4.2)
3D matrix (possibly sparse) $F_k^-(\omega, \beta_1, \beta_2)$
4D matrix (possibly sparse) $F_k^\pm(\omega_1, \omega_2, \beta_1, \beta_2)$
Solver
3D / 4D Interpolation at each mth summation step (Section 6.2)
$Z(\omega)$ $\Phi(\omega)$ Limits
$\sum$ Inner sum
$F_k^\pm(\omega_m, \omega_n, \beta_m, \beta_n)$
$F_{ex\ k}^{\pm(2)}$

Figure 5.1: *Generalized overview calculation of second order forces. Here $Z(\omega)$ is the combination complex wave as a function of frequency and $\Phi(\omega)$ is the associated wave direction heading for each frequency. The 3D and 4D arrays containing the wave force transfer function are interpolated at each step within the innermost summation containing the $F_k^\pm$ terms, or during assembly of the matrix for the FFT.*

or four-dimensional linear interpolation subroutines (see Chapter 6 for details on the interpolation). The interpolation of $F_k^\pm$ occurs within each step of the summation or during the assembly of the FFT array. This minimizes the memory storage requirements somewhat.

5.2 Frequency range

Add discussion of Nyquist frequency and its effects – see e-mail and discussion within the code

The number of timesteps used in the *Waves* module is determined by the total time, t_{max}, and timestep, Δt, specified for wave information (`WaveTMax` and `WaveDT` in the HydroDyn input file and code). The number of timesteps (`NStepWave`) is then set to at least $N \geq \left(\frac{t_{\text{max}}}{\Delta t}\right)$ such that N is even and $N/2$ is a product of small numbers (this is advantageous in computing the FFT). The number of timesteps also limits the number of frequencies present in the wave information to $N/2+1$ frequencies with a maximum frequency of $1/(2\Delta t)$ in Hz (the Nyquist frequency). The maximum wave frequency is therefore $\omega_{\text{max}} = N/2 \cdot \Delta\omega$ where ω is in radians. Due to the specifics of the FFT used (see Section 6.1, the number of frequencies stored in the wave information is $N/2 + 1$ where only positive frequencies are used (values for $\omega < 0$ are derived from the values for $\omega > 0$). The full expression for N in can be written as

$$N = \frac{t_{\text{max}}}{\Delta t} = \frac{2\omega_{\text{max}}}{\Delta\omega} = \frac{2\pi}{\Delta t \Delta\omega}. \tag{5.1}$$

Three sets of wave frequency cutoffs are applied to the wave conditions. The first order frequency range is bounded by the limits ω_{lo} and ω_{hi} (set in the HydroDyn input file by `WvLowCOff` and `WvHiCOff`). These limits are imposed during generation of the wave frequencies in the *Waves* module by setting the amplitudes of $Z(\omega) = 0$ for all ω outside the limits.

The second order wave frequency limits for all three difference methods are given by $\omega_{\text{lo-d}}$ and $\omega_{\text{hi-d}}$ (`WvLoCoQTFd` and `WvHiCoQTFd`). In the case of the mean-drift and Newman's approximation methods where only the diagonal elements (where $\omega_1 = \omega_2$) are considered, the low difference frequency cutoff will likely not have any impact since it will be rare that $\omega_{\text{lo-d}} > \omega_{\text{lo}}$ (a warning will be issued if this does occur).[1] The high frequency cutoff will likely be set so that $\omega_{\text{hi-d}} < \omega_{\text{hi}}$, so this will be important to impose (a warning will be issued if this is not true).

The second order wave frequency limits for the full second order QTFs are then set for the difference QTF and sum QTF methods individually. For the difference QTF method these limits are given by $\omega_{\text{lo-d}}$ and $\omega_{\text{hi-d}}$ (`WvLowCOffD` and `WvHiCOffD`). For the sum QTF method these limits are given by $\omega_{\text{lo-s}}$ and $\omega_{\text{hi-s}}$ (`WvLowCOffS` and `WvHiCOffS`). These limits are applied during the summations of the $a_m a_m F(\omega_m, \omega_m)$ and $H_{\mu^\pm}$ terms.

5.3 Wave Amplitude

The complex wave amplitude in frequency space, $Z(\omega)$, provided by the *Waves* module contains an extra $N/2$ factor on the amplitude. This factor is not present in the equations given in Ref. 1. The relationship between the amplitude provided by the *Waves* module, A_m and the amplitude used in the paper, a_m, is

$$A_m = \frac{N}{2} a_m. \tag{5.2}$$

In the initialization of the *WAMIT2* module, the values stored for the wave elevation from the *Waves* module are divided by $N/2$ before being stored in the `aWaveElevC0` variable.

5.4 Difference Frequency Force

There are three methods for calculating the difference frequency force: mean drift, Newman's approximation, and through the full QTF. Each method has its advantages, and has different data requirements. In this section, we explore the simplified equations used in the *WAMIT2* module, the data requirements, and the data processing steps for each of the three methods.

[1] The mean drift and Newman's approximation will automatically be zeroed for frequencies less than ω_{lo} since they contain a_m terms for each $\omega_m = \omega_m$ diagonal term.

5.4.1 Mean-Drift Method

This term arises from the quadratic interactions of the first order problem, and can therefore be calculated without requiring solutions to the second order potential. This equation is identical to the first part of the difference QTF equation, Equation (5.13), where just the diagonal elements of the QTF are used. This is the simplest of the three difference frequency methods presented here. The single summation equation is given by

$$F^{\text{-}(2)}_{\text{ex}\ k} = \Re\left(\sum_{m=1}^{N/2} a_m a_m^* F^{\text{-}}_k(\omega_m, \beta_m)\right) \qquad \text{for} \quad k = 1, 2, \ldots, 6, \tag{5.3}$$

where k indicates the index to the load component, $F^{\text{-}(2)}_{\text{ex}\ k}$ is the resulting time independent mean drift force, and a_m and a_m^* are the complex wave amplitude and its complex conjugate for the m^{th} frequency. Note the lack of time dependence in this equation: the mean drift is the average drift force over the entire simulation. Note that $F^{\text{-}}_k(\omega_m, \beta_m)$ is the dimensionalized real valued diagonal of the QTF read from the WAMIT file and interpolated for the m^{th} wave frequency. Note that the $\Delta\omega$ term is necessary since this is a numerical integral. Note also that the summation starts at $m = 1$. The value of a_0 is exactly 0, so it does not need to be included.

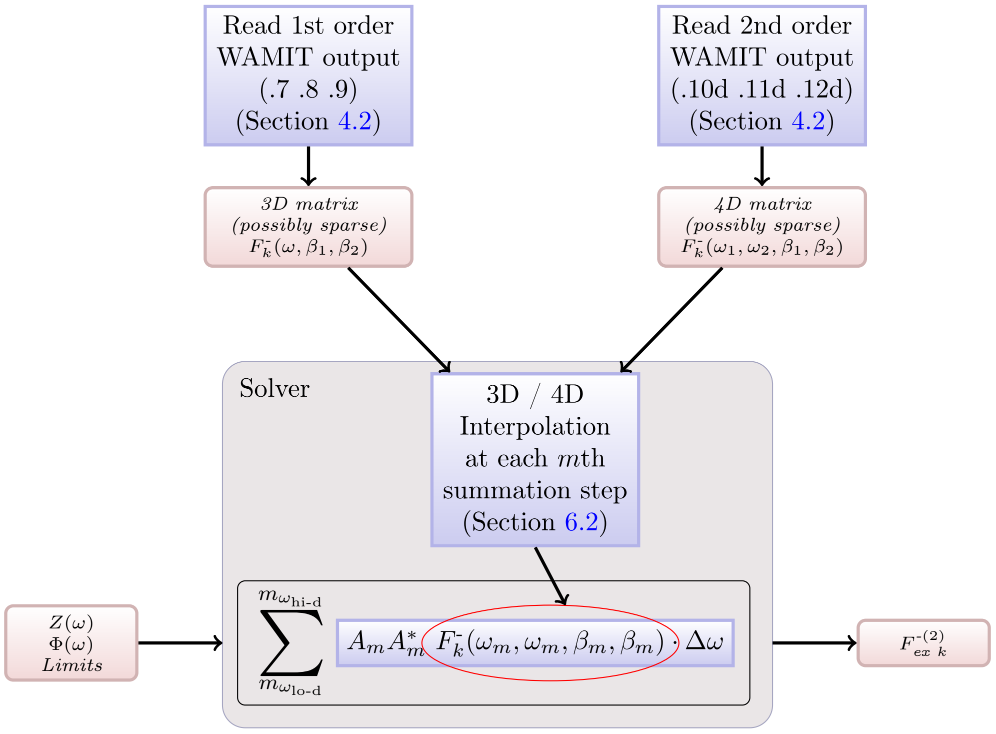
Read 1st order WAMIT output (.7 .8 .9) (Section 4.2)
Read 2nd order WAMIT output (.10d .11d .12d) (Section 4.2)
3D matrix (possibly sparse) $F^{\text{-}}_k(\omega, \beta_1, \beta_2)$
4D matrix (possibly sparse) $F^{\text{-}}_k(\omega_1, \omega_2, \beta_1, \beta_2)$
Solver
3D / 4D Interpolation at each mth summation step (Section 6.2)
$\sum_{m_{\omega_{\text{lo-d}}}}^{m_{\omega_{\text{hi-d}}}} A_m A_m^* F^{\text{-}}_k(\omega_m, \omega_m, \beta_m, \beta_m) \cdot \Delta\omega$
$Z(\omega)$ $\Phi(\omega)$ Limits
$F^{\text{-}(2)}_{ex\ k}$

Figure 5.2: *Flow diagram for the mean drift calculation method. The mean drift equation only involves the diagonal terms where the frequencies $\omega_1 = \omega_2$, and the wave directions $\beta_1 = \beta_2$. The WAMIT output files are read in and arranged in either a 3D or 4D array and interpolated at each step in the summation. See text for how to solve the equation. See Chapter 4 for requirements on which WAMIT output files can be used.*

Solving the Mean-Drift Equations

As shown in Figure 5.2, the data is stored in either 3D or 4D arrays depending on the file type used. This is handled by the WAMIT output file reading subroutine, `Read_DataFiles`, within the *WAMIT2* module. At each step in the summation of the m^{th} term, a call is made to the 3D or 4D interpolation algorithm to find the value of $F_k^{-}(\omega_m, \beta_m)$ corresponding to the $Z(\omega_m)$ term in the complex wave amplitude a_m. The limits of $\omega_{\text{lo-d}} \leq \omega_m \leq \omega_{\text{hi-d}}$ are imposed during the summation with values outside this range set to zero.

For multi-directional waves where the equal energy discretization is used, each frequency has a single wave direction associated with it. Since the mean drift force calculation only involves summing over terms involving only a single frequency at a time, only a single wave direction is involved at each step. If all the diagonal elements where $\omega_1 = \omega_2$ and $\beta_1 = \beta_2$ were present in the F_k^{-} arrays were present, it would be possible to simplify the interpolation required to two dimensional interpolation. However, since this calculation is performed only at initialization, the calculation penalty for performing a full 3D or 4D interpolation is not as severe as it would be if it were performed at each timestep of the simulation. Therefore we are choosing to ignore this in favor of simplifying the code implimentation.

5.4.2 Newman's Approximation Method

Newman's original approximation:

$$F^{-(2)}_{\text{ex}\ k} \approx \left[\Re\left(\sum_{m=1}^{N/2} a_m \sqrt{2F_k^-(\omega_m,\omega_m)}\cdot \mathrm{e}^{i\omega_m t}\right)\right]^2\Bigg|_{F_k^-(\omega_m,\omega_m)>0} - \left[\Re\left(\sum_{m=1}^{N/2} a_m \sqrt{-2F_k^-(\omega_m,\omega_m)}\cdot \mathrm{e}^{i\omega_m t}\right)\right]^2\Bigg|_{F_k^-(\omega_m,\omega_m)<0} \quad \text{for} \quad k=1,2,\ldots,6, \tag{5.4}$$

where k indicates the index to the load component, and a_m is the complex wave amplitude for the m^{th} frequency. Note that F_k^- is the complex valued transfer function read from the WAMIT file and interpolated. This equation is only valid for uni-directional sea states.

In evaluating the Newman approximation, it must be remembered that the purpose of it is to estimate the off-diagonal ($\omega_1 \neq \omega_2$) values from the diagonal values. If the QTF has large off diagonal components (where $\omega_m \neq \omega_n$), the results will not agree well with the full difference QTF calculations since these elements will be under estimated.

A revised form of Newman's approximation is provided by Standing *et. al.* that is valid for multidirectional sea states. For multidirectional sea states where each frequency only has on direction assocatied with it, such as with the equal energy discretization, this is equivalent to

$$F^{-(2)}_{\text{ex}\ k} \approx \left|\sum_{m=1}^{N/2} a_m \sqrt{F_k^-(\omega_m,\omega_m)}\cdot \mathrm{e}^{i\omega_m t}\right|^2\Bigg|_{F_k^-(\omega_m,\omega_m)>0} - \left|\sum_{m=1}^{N/2} a_m \sqrt{-F_k^-(\omega_m,\omega_m)}\cdot \mathrm{e}^{i\omega_m t}\right|^2\Bigg|_{F_k^-(\omega_m,\omega_m)<0} \quad \text{for} \quad k=1,2,\ldots,6, \tag{5.5}$$

This equation has been implimented as a Fourier sum rather than IFFT. It turns out that this implimentation is exactly the same speed as doing the full DiffQTF (within 3%).

5.4.2.1 Mean drift and Newman's approximation

The mean drift term is already included in the Newman's approximation, so there is no need to add it to the result. This can be proven by expanding out the squares of the summations and collecting all terms where $m = n$ together. These collected terms can be written as the mean drift term. For this proof, it is helpful to remember that for a a given complex value, $C = a + ib$, we can write:

$$(|C|)^2 = \left(\sqrt{a^2+b^2}\right)^2 = a^2+b^2 = (a+ib)(a-ib) = C\cdot C^* \tag{5.6}$$

where C^* is the complex conjugate of C.

Taking Equation (5.5) and expanding out the terms using Equation (5.6)

$$F^{-(2)}_{\text{ex}\ k} \approx \left(\sum_{m=1}^{N/2} a_m \sqrt{F_k^-(\omega_m,\omega_m)}\cdot \mathrm{e}^{i\omega_m t}\right)\left(\sum_{m=1}^{N/2} a_m^* \sqrt{F_k^{-\ *}(\omega_m,\omega_m)}\cdot \mathrm{e}^{-i\omega_m t}\right)\Bigg|_{F_k^-(\omega_m,\omega_m)>0} - \left(\sum_{m=1}^{N/2} a_m \sqrt{-F_k^-(\omega_m,\omega_m)}\cdot \mathrm{e}^{i\omega_m t}\right)\left(\sum_{m=1}^{N/2} a_m^* \sqrt{-F_k^{-\ *}(\omega_m,\omega_m)}\cdot \mathrm{e}^{-i\omega_m t}\right)\Bigg|_{F_k^{-\ *}(\omega_m,\omega_m)<0}. \tag{5.7}$$

Looking at just the first term ($F_k^-(\omega_m,\omega_m) > 0$) and remembering that $F_k^-(\omega_m,\omega_m) = F_k^{-\,*}(\omega_m,\omega_m)$, we can expand all the crossmultiplied terms, collect the terms where $m = n$, and group the other terms where $m \neq n$. This yields for the first term

$$\begin{aligned}\text{Term 1} = {} & a_1a_1^*F_k^-(\omega_1,\omega_1)\mathrm{e}^{i(\omega_1-\omega_1)t} + a_2a_2^*F_k^-(\omega_2,\omega_2)\mathrm{e}^{i(\omega_2-\omega_2)t} + \dots \\ & + a_{N/2}a_{N/2}^*F_k^-(\omega_{N/2},\omega_{N/2})\mathrm{e}^{i(\omega_{N/2}-\omega_{n/2})t} \\ & + O_{m\neq n}\left(\left(\sum_{m=1}^{N/2} a_m\sqrt{F_k^-(\omega_m,\omega_m)}\cdot\mathrm{e}^{i\omega_m t}\right)\left(\sum_{m=1}^{N/2} a_m^*\sqrt{F_k^{-\,*}(\omega_m,\omega_m)}\cdot\mathrm{e}^{-i\omega_m t}\right)\right) \\ = {} & \sum_{m=1}^{N/2} a_ma_m^*F_k^-(\omega_m,\omega_m) \\ & + O_{m\neq n}\left(\left(\sum_{m=1}^{N/2} a_m\sqrt{F_k^-(\omega_m,\omega_m)}\cdot\mathrm{e}^{i\omega_m t}\right)\left(\sum_{m=1}^{N/2} a_m^*\sqrt{F_k^{-\,*}(\omega_m,\omega_m)}\cdot\mathrm{e}^{-i\omega_m t}\right)\right). \end{aligned} \tag{5.8}$$

The second term is very similar to the first term. Keeping the negative sign with the term, when expanded and multiplied together it yields

$$\begin{aligned}\text{Term 2} = {} & -a_1a_1^*(-F_k^-(\omega_1,\omega_1))\mathrm{e}^{i(\omega_1-\omega_1)t} + -a_2a_2^*(-F_k^-(\omega_2,\omega_2))\mathrm{e}^{i(\omega_2-\omega_2)t} + \dots \\ & - a_{N/2}a_{N/2}^*(-F_k)^-(\omega_{N/2},\omega_{N/2})\mathrm{e}^{i(\omega_{N/2}-\omega_{n/2})t} \\ & - O_{m\neq n}\left(\left(\sum_{m=1}^{N/2} a_m\sqrt{-F_k^-(\omega_m,\omega_m)}\cdot\mathrm{e}^{i\omega_m t}\right)\left(\sum_{m=1}^{N/2} a_m^*\sqrt{-F_k^{-\,*}(\omega_m,\omega_m)}\cdot\mathrm{e}^{-i\omega_m t}\right)\right) \\ = {} & \sum_{m=1}^{N/2} a_ma_m^*F_k^-(\omega_m,\omega_m) \\ & + O_{m\neq n}\left(\left(\sum_{m=1}^{N/2} a_m\sqrt{-F_k^-(\omega_m,\omega_m)}\cdot\mathrm{e}^{i\omega_m t}\right)\left(\sum_{m=1}^{N/2} a_m^*\sqrt{-F_k^{-\,*}(\omega_m,\omega_m)}\cdot\mathrm{e}^{-i\omega_m t}\right)\right). \end{aligned} \tag{5.9}$$

Combining the first terms from Equation (5.8) and Equation (5.9) where the first one contains all positive F^- terms and the second contains all negative F^- terms, we arrive at the equation for the mean drift given in Equation (5.3)

$$\sum_{m=1}^{N/2} a_ma_m^*F_k^-(\omega_m,\omega_m)\Bigg|_{F_k^{-\,*}(\omega_m,\omega_m)>0} + \sum_{m=1}^{N/2} a_ma_m^*F_k^-(\omega_m,\omega_m)\Bigg|_{F_k^{-\,*}(\omega_m,\omega_m)<0} = \sum_{m=1}^{N/2} a_ma_m^*F_k^-(\omega_m,\omega_m). \tag{5.10}$$

5.4.2.2 Standing's equation

Starting with Equation (5.7), we can substitute in for the $-F_k^- = \left|F_k^-\right|$ within the square root for when $F_k^{-\;*}(\omega_m,\omega_m) < 0$ and $F_k^- = \left|F_k^-\right|$ within the square root for when $F_k^{-\;*}(\omega_m,\omega_m) > 0$ to yield

$$\begin{aligned} F_{\text{ex}\;k}^{-(2)} \approx & \left(\sum_{m=1}^{N/2} a_m \sqrt{\left|F_k^-(\omega_m,\omega_m)\right|} \cdot \mathrm{e}^{i\omega_m t}\right)\left(\sum_{m=1}^{N/2} a_m^* \sqrt{\left|F_k^-(\omega_m,\omega_m)\right|} \cdot \mathrm{e}^{-i\omega_m t}\right)\Bigg|_{F_k^-(\omega_m,\omega_m)>0} \\ & - \left(\sum_{m=1}^{N/2} a_m \sqrt{\left|F_k^-(\omega_m,\omega_m)\right|} \cdot \mathrm{e}^{i\omega_m t}\right)\left(\sum_{m=1}^{N/2} a_m^* \sqrt{\left|F_k^-(\omega_m,\omega_m)\right|} \cdot \mathrm{e}^{-i\omega_m t}\right)\Bigg|_{F_k^{-\;*}(\omega_m,\omega_m)<0} . \end{aligned} \tag{5.11}$$

Now that F_k^- is treated in such a way that the sign of it is not kept within the square root, we can collect the first and second term together in Equation (5.11) by introducing the sgn function as

$$\begin{aligned} F_{\text{ex}\;k}^{-(2)} \approx & \left(\sum_{m=1}^{N/2} \operatorname{sgn}\left(F_k^-\right) a_m \sqrt{\left|F_k^-(\omega_m,\omega_m)\right|} \cdot \mathrm{e}^{i\omega_m t}\right)\left(\sum_{m=1}^{N/2} a_m^* \sqrt{\left|F_k^-(\omega_m,\omega_m)\right|} \cdot \mathrm{e}^{-i\omega_m t}\right)\Bigg|_{F_k^-(\omega_m,\omega_m)>0} \\ & + \left(\sum_{m=1}^{N/2} \operatorname{sgn}\left(F_k^-\right) a_m \sqrt{\left|F_k^-(\omega_m,\omega_m)\right|} \cdot \mathrm{e}^{i\omega_m t}\right)\left(\sum_{m=1}^{N/2} a_m^* \sqrt{\left|F_k^-(\omega_m,\omega_m)\right|} \cdot \mathrm{e}^{-i\omega_m t}\right)\Bigg|_{F_k^{-\;*}(\omega_m,\omega_m)<0} \\ = & \left(\sum_{m=1}^{N/2} \operatorname{sgn}\left(F_k^-\right) a_m \sqrt{\left|F_k^-(\omega_m,\omega_m)\right|} \cdot \mathrm{e}^{i\omega_m t}\right)\left(\sum_{m=1}^{N/2} a_m^* \sqrt{\left|F_k^-(\omega_m,\omega_m)\right|} \cdot \mathrm{e}^{-i\omega_m t}\right) \\ = & \left(\sum_{m=1}^{N/2} \operatorname{sgn}\left(F_k^-\right) \widetilde{a_m} \sqrt{\left|F_k^-(\omega_m,\omega_m)\right|} \cdot \mathrm{e}^{i\omega_m t-\phi_m}\right)\left(\sum_{m=1}^{N/2} \widetilde{a_m} \sqrt{\left|F_k^-(\omega_m,\omega_m)\right|} \cdot \mathrm{e}^{-i\omega_m t+\phi_m}\right) \end{aligned} \tag{5.12}$$

which is Standings version of the equation with $\widetilde{a_m}\; e^{-\phi_m} = a_m$ (Standing carries the phase term of the wave amplitude within the exponential).

5.4.3 Difference QTF Method

$$F_{\text{ex }k}^{(-)(2)} = \Re\left(\sum_{m=1}^{N/2} a_m a_m^* F_k^-(\omega_m, \omega_m) + 2 \cdot \sum_{\mu^-=1}^{N/2-1} H_{\mu^-} \mathrm{e}^{i(\omega_{\mu^-})t}\right), \qquad \text{for} \quad k = 1, 2, \ldots, 6 \tag{5.13}$$

$$\text{where} \quad H_{\mu^-} = \frac{1}{2} \sum_{n=0}^{N/2-\mu^-} a_{\mu^-+n} a_n^* F_k^-(\omega_{\mu^-+n}, \omega_n), \qquad \text{for} \quad 1 \leq \mu^- \leq N-1 \tag{5.14}$$

In the first term, the summation starts at $m = 1$. The value of a_0 is exactly 0, so it does not need to be included.

5.5 Summation Frequency Force

There is only one method for calculating the second-order forces from the sum frequency.

5.5.1 Summation QTF Method

$$F_{\text{ex}\ k}^{(+)(2)} = \Re\left(\sum_{m=1}^{\lfloor N/4\rfloor} a_m a_m F_k^{+}(\omega_m,\omega_m)\mathrm{e}^{2\cdot i\omega_m t} + 2\sum_{\mu^+=2}^{N} H_{\mu^+}\mathrm{e}^{i(\omega_{\mu^+-1})t}\right), \quad \text{for} \quad k=1,2,\dots,6 \tag{5.15}$$

$$\text{where} \quad \begin{cases} H_{\mu^+} = \displaystyle\sum_{n=1}^{\left\lfloor\frac{\mu^+-1}{2}\right\rfloor} a_n a_{\mu^+-n} F_k^{+}(\omega_n,\omega_{\mu^+-n}), & \text{for} \quad 2\le\mu^+\le N/2+1 \\ H_{\mu^+} = \displaystyle\sum_{n=\mu^+-N}^{\left\lfloor\frac{\mu^+-1}{2}\right\rfloor} a_n a_{\mu^+-n} F_k^{+}(\omega_n,\omega_{\mu^+-n}), & \text{for} \quad N/2+2\le\mu^+\le N \end{cases} \tag{5.16}$$

where $\mu^+ = m+n$, $\lfloor x\rfloor$ represents the floor function given by

$$\lfloor x\rfloor \equiv \max\{m\in\mathbb{Z}|m\le x\}.$$

In the first term of Equation (5.15), the exponential term is $2\cdot i\omega_m t$. This means that in populating the array that the IFFT uses for calculating the time series of this term, the frequency of the m^{th} term is $2\cdot\omega_m$. So in the numerical implimentation, every other frequency is populated in the frequency domain data.

The second term is an IFFT over the full range of sum frequencies from $\omega=0$ to $\omega=2\omega_{\max}$. The IFFT contains twice as many terms as any of the IFFTs used in the other methods. This results in a finer resolution in the resulting time series with timesteps of $1/2\Delta t$. While this in and of itself is not problematic, it is a problem when considering that the highest frequencies in this IFFT are above the Nyquist frequency. The Nyquist frequency, as given in Section 6.1, is the highest frequency from the sampling theorem that is possible for a given timestep. Any higher frequencies will be lost when we report the force time series back to the calling code (assuming we keep only every other). So, rather than calculate the full IFFT over the extended range, we will only calculate the second term for the first $N/2$ terms up to the Nyquist frequency. This simplifies the second term so that Equation (5.15) becomes

$$F_{\text{ex}\ k}^{(+)(2)} = \Re\left(\sum_{m=1}^{\lfloor N/4\rfloor} a_m a_m F_k^{+}(\omega_m,\omega_m)\mathrm{e}^{2\cdot i\omega_m t} + 2\sum_{\mu^+=2}^{N/2} H_{\mu^+}\mathrm{e}^{i(\omega_{\mu^+-1})t}\right), \quad \text{for} \quad k=1,2,\dots,6 \tag{5.17}$$

$$\text{where} \quad \left\{ H_{\mu^+} = \sum_{n=1}^{\left\lfloor\frac{\mu^+-1}{2}\right\rfloor} a_n a_{\mu^+-n} F_k^{+}(\omega_n,\omega_{\mu^+-n}), \quad \text{for} \quad 2\le\mu^+\le N/2 \right. \tag{5.18}$$

Chapter 6

Algorithms

This chapter will cover the various algorithms, including interpolation, that are required for either preparing data or in the evaluation of the equations outlined in Chapter 5.

The reason for the interpolation is that the number of frequencies given in the WAMIT output files (on the order of tens of frequencies) is not likely going to correspond to the number of wave frequencies actually used by HydroDyn (on the order of hundreds to thousands), nor does the WAMIT output necessarily have to be equally discretized or even complete. The WAMIT output files may be very sparcely populated. So, it is necessary to interpolate in order to find the missing ones.

6.1 FFT and IFFT

The FFT (or discrete Fourier transform – DFT) and inverse FFT used in HydroDyn are found in the FFTPACK version 4.1 from UCAR/NCAR. For a given discretized function, for example the complex wave form $Z[k]$ in the frequency domain, the inverse fourier transform to the time domain can be written as:

$$z(t_n) = \frac{1}{N} \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} Z[k] e^{i\omega_k t_n} = \Re \left\{ \sum_{k=1}^{N'} a_k e^{iw_k t_n} \right\}, \tag{6.1}$$

where N is given in Equation (5.1) as

$$N = \frac{2\pi}{\Delta t \Delta\omega} = \frac{t_{\max}}{\Delta t} = \frac{2\omega_{\max}}{\Delta\omega} = 2(N'+1). \tag{6.2}$$

In Equation (6.1), the expression for the first summation is what is used within HydroDyn and the second summation expression is used in some of Tiago's writings. The relationship between $Z[k]$ and a_k can be written as

$$Z[k] = \begin{cases} \dfrac{Na_k}{2} & k = 1 \ \ldots \ N/2-1 \\ 0 & k = 0 \text{ and } k = N/2 \\ \dfrac{Na^*_{|k|}}{2} & k = -N/2+1 \ \ldots \ -1 \end{cases} \tag{6.3}$$

where a^* is the complex conjugate of a.

6.1.1 Numerical Evaluation of IFFT

In the evaulation of Equation (6.1) in HydroDyn to yield the wave height as a function of time, the IFFT is evaluated as

$$z(t_n) = \frac{1}{N} \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} Z[k] e^{i\omega_k t_n} = \text{IFFT}\left(Z[k]\right) \tag{6.4}$$

where the IFFT is only evaluated over $k = 0 \ldots N/2$ (the negative frequencies are evaluated internally following the relationships in Equation (6.3)). The normalization constant of $1/N$ is also handled by the IFFT subroutines and is set by the initialization of the IFFT.

There are some constraints imposed on what N can be because of the IFFT solver used. N must be even, and preferably a product of small prime numbers for speed. Additionally, $Z[k = 0] = 0$ and $Z[k = N/2] = 0$ must be specified.

6.1.2 $Z[k]$ in HydroDyn

In HydroDyn the complex wave form in frequency space, $Z[k]$, is given as

$$Z[k] = W[k] \sqrt{\frac{2\pi}{\Delta t} S_\zeta^{\text{2-sided}}(\omega_k)} = W[k] \sqrt{N \Delta\omega S_\zeta^{\text{2-sided}}(\omega_k)} \tag{6.5}$$

where

$$W[k] = \sqrt{\frac{N}{2}} \sqrt{-2\ln\left(U_1[k]\right)} e^{i2\pi U_2[k]}, \tag{6.6}$$

the DFT of gaussian white noise using the Box-Muller method, and $S_\zeta^{\text{2-sided}}(\omega_k)$ is the two sided power spectral density (PSD) of the wave elevation per unit time.[1]

6.2 Interpolation

Four interpolation algorithms are required for the *WAMIT2* module: two three-dimensional and two four-dimensional interpolations. For each set of 3D and 4D interpolation algorithms, a linear interpolation for full arrays and a linear interpolation for sparse arrays are needed. Due to the complexity of implimenting an interpolation over sparse arrays and time constraints in the development schedule, a placeholder will be created for it with an error message stating that an interpolation scheme for sparse arrays is not implimented at this time. The WAMIT output file reading algorithm is developed in such a way that an unordered sparse array can be read in and stored (see Section 4.2).

The first order wave forces calculated within the *WAMIT* module is interpolated with a linear method. In light of this and considering time constraints on the development, we will use linear interpolation algorithms for now. If time permits, we might investigate possibilities for other interpolation algorithms that will produce smooth surfaces is cubic interpolation (and smoothly continuous derivatives) or better allow for sparse data.

6.2.1 3D Interpolation

6.2.1.1 Full array interpolation

A three-dimensional linear interpolation routine is available in the *InflowWind* module. This routine was written specifically for the full field wind files, so it will require some modification to generalize it.

[1]Note that Tiago uses $a_k = \sqrt{2\Delta\omega S_\zeta^{\text{1-sided}}(\omega_k)} e^{i2\pi U_k}$ which is a simplification where $U_k = U_2[k]$ and $\sqrt{-2\ln\left(U_1[k]\right)} = \sqrt{2}$.

6.2.1.2 Sparse array interpolation

Due to time constraints in the development schedule, a placeholder subroutine will be created that tells the user that this is a currently unavailable feature. The user can then use an external data manipulation program to do the interpolation on their WAMIT output to create a full array (that can be unordered) that can be read in. The WAMIT output file reading algorithm will accomodate the reading either a full array (both the upper and lower triangle of the QTF) or partial array (upper half only, or a mix of upper and lower). This routine will expect a mask array (boolean?) of identical size to the sparse array that indicates which elements of the data array are missing.

By creating this placeholder subroutine, we give ourselves the option of creating this interpolation scheme as time permits with the ability to handle a limited sparseness of the QTF array (*i.e.* no more than a two step gap in any dimension).

6.2.2 4D Interpolation

6.2.2.1 Full array interpolation

At present we do not have any four-dimensional interpolation routines. A four-dimensional linear interpolation should be fairly simple to extend from the three-dimensional one.

6.2.2.2 Sparse array interpolation

See Section 6.2.1.2.

Chapter 7

Integration of the WAMIT2 module within HydroDyn

In order to integrate the WAMIT2 module into HydroDyn, several changes need to be made to the HydroDyn input file and to the file reading and parsing routines within the *HydroDyn_ Input.f90* file. Additional changes to the input file are required for multi-directional waves (see Chapters 3 and 8).

```
!----------------------  WAMIT 2nd order PLATFORM forces  -------------------
 8        MnDrift          ! Mean drift forces computed from WAMIT file
 0        NewmanApp        ! Slow drift forces computed with Newman's approximation from WAMIT file
 0        DiffQTF          ! Full Difference-Frequency forces computed with full QTF's from WAMIT file
 0        SumQTF           ! Full Sum-Frequency forces computed with full QTF's from WAMIT file
```

Table 7.1: *New section for the HydroDyn input file for the second order forces calculated by the WAMIT2 module.*

The additional second order waves information that needs to be added into the HydroDyn input file is give in Table 7.1. This is inserted between the sections marked FLOATING PLATFORM and FLOATING PLATFORM FORCE FLAGS. To decide which force components are calculated, the HydroDyn_Input copies the component direction information from the FLOATING PLATFORM FORCE FLAGS section to the WAMIT2%InitInput. The WAMIT2_Init subroutine decides which of the force components can be calculated based on the information available within the WAMIT output file and method chosen.

Add in section here describing what combinations of MnDrift, NewmanApp, DiffQTF, and SumQTF are allowed – table perhaps?

Add in description of the inputs with more detail about when used etc.

Chapter 8

Integration of the Waves2 module within HydroDyn

In order to integrate the Waves2 module into HydroDyn, several changes need to be made to the HydroDyn input file and to the file reading and parsing routines within the *HydroDyn_ Input.f90* file. Additional changes to the input file are required for multi-directional waves and the *WAMIT2* module (see Chapters 3 and 7).

```
---------------------- 2ND-ORDER WAVES ----------------------------------------
 FALSE   WvMnDrift    - Mean-drift               2nd-order wave kinematics (switch) [only one of WvMnDrift or WvDiffQTF can be TRUE]
 FALSE   WvDiffQTF    - Full difference-frequency 2nd-order wave kinematics (switch) [only one of WvMnDrift or WvDiffQTF can be TRUE]
 FALSE   WvSumQTF     - Full summation-frequency  2nd-order wave kinematics (switch)
 0       WvLowCOffD   - Low  frequency cutoff used in the difference-frequencies (rad/s) [Only used with a difference-frequency method]
 3.5     WvHiCOffD    - High frequency cutoff used in the difference-frequencies (rad/s) [Only used with a difference-frequency method]
 0.1     WvLowCOffS   - Low  frequency cutoff used in the summation-frequencies  (rad/s) [Only used with a summation-frequency  method]
 3.5     WvHiCOffS    - High frequency cutoff used in the summation-frequencies  (rad/s) [Only used with a summation-frequency  method]
```

Table 8.1: *New section for the HydroDyn input file for the second order forces calculated by the Waves2 module.*

Add in description of the inputs with more detail about when used etc.

Chapter 9

WAMIT2 Module Architecture

This section is copied from the old writeup. **It has not been updated.**

Add in the registry infromation as needed

Figure 1 shows the development architecture of the WAMIT2 module for HydroDyn. Once it has been developed, this module will eventually be incorporated into HydroDyn. During development, it will interface with its driver routine which will mimic the interface of the HydroDyn module. Because there is ongoing development in the existing HydroDyn modules, a copy of the modularized Waves sub-module (Waves2) will be used and converted to handle bi-directional waves. This will be merged back into the Waves module later.

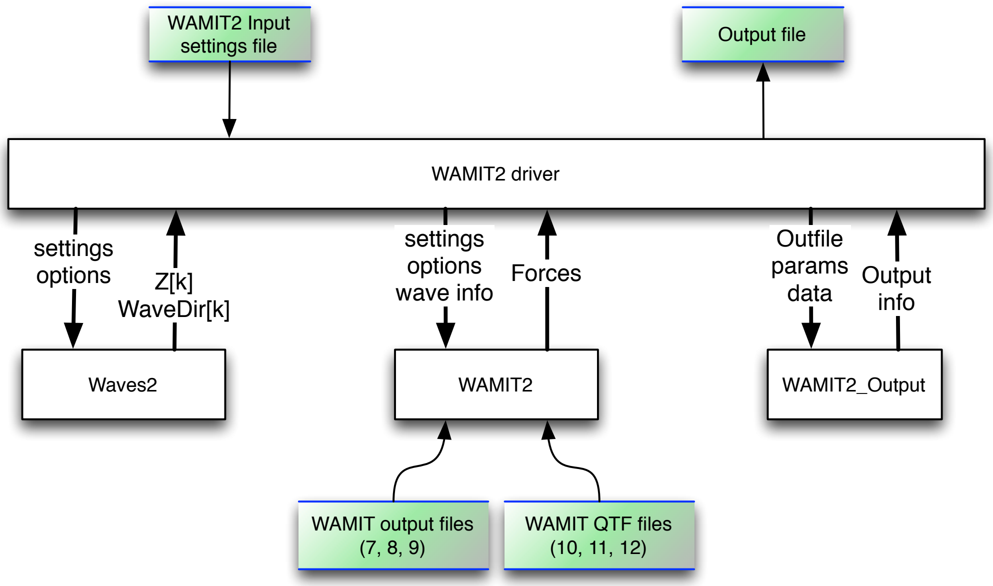

Figure 9.1: *Information flow to and from the WAMIT2 module. The green boxes indicate information or files passed into or out of the module.*

Bibliography

[1] T. Duarte, A. JNA. Sarmento, and J. Jonkman. Effects of second-order hydrodynamic forces on floating offshore wind turbines. Technical Report CP-5000-60966, NREL, Golden, Colorado, Apr. 2014.

[2] *WAMIT User Manual*. Chestnut Hill, MA.