Assessment and validation of bathymetric inversion techniques using synthetic waves

Thesis submitted for the Bacherlor's Degrees in

Engineering Physics

Telecommunications Technologies and Services Engineering

presented by

Pau Luque Lozano

directed by

Daniel Calvete Manrique

Francesca Ribas Prats

Gonzalo Simarro Grande

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de Formació Interdisciplinària Superior

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

February, 2019

Summary

Bathymetric inversion using video images is a new and promising technique in order to monitor beach morphology. In this TFG, a Boussinesq model called FUNWAVE and a linear wave propagation routine will be used to create synthetic wave propagation over different bathymetries. Two bathymetric inversion methodologies, cBathy and uBathy, will be subsequently applied to retrieve the bathymetry, comparing their results and analysing their sensitivity to different aspects, in order to understand and improve their performance.

Keywords Bathymetry, bathymetric inversion, cBathy, uBathy, FUNWAVE.

Index

1 . Introduction

The coast is one of the most important areas of the planet, due to its environmental, touristic, commercial, logistic and demographic importance. The beaches and the nearshore waters that form it are quite dynamical, undergoing multiple and complex changes. The states that characterize the phenomena of those systems are related to their distribution of depth, which is called the bathymetry (Wright and Short [1984]). In fact, their evolution and their behaviour are dependent depend on it (Coco et al. [2007]). Therefore, the bathymetry is a required input for predictions, coastal management decisions and scientific investigations upon those systems.

The usual method to measure the bathymetry involves in place surveying using direct contact or a ship-mounted sonar (echo sounding). More recently, other types of vehicles and technologies, such as jet-skis (Dugan et al. [2001], echo sounding), amphibious buggies (Birkemeier and Mason [1984], direct contact) and aircraft (Irish and Lillycrop [1999], LIDAR), have been used. There have been also attempts to use air-born or space-born colorimetry to obtain the bathymetry (Lyzenga et al. [2006]) (works only in very clear waters). However, all these methods are expensive and time-consuming, resulting in low spatiotemporal resolutions: it is usual to perform just one survey per year, and only measuring some transects out of the actual nearshore area.

To overcome this, a series of alternative methods have been proposed, using remote observations to provide spatially denser and more frequent bathymetries, with a lower cost. There was an initial attempt during WW1 and WW2 (Williams [1947]), using images of beaches (obtained from aircraft) to guess the bathymetry of enemy-held coasts. They were based on the evolution of the water-line with the tides, and the dependence of the wavelength and the phase speed of waves with depth for near waters, given by the Airy dispersion equation in the small amplitude limit.

$$h = \frac{1}{k} \operatorname{arctanh}\left(\frac{w^2}{gk}\right) \tag{1.1}$$

Here, h is the water depth, k is the wavenumber of the waves, ω stands for the waves angular frequency and g symbolizes the gravity acceleration. This method has been called bathymetry inversion, The performance of these initial attempts was very very limited. However, new sensing apparatus, mainly video cameras (Lippmann and Holman [1989]) and X-band radar (Grilli [1998]), have provided a huge advancement in the remote sensing techniques for bathymetry inversion in the recent decades. This instruments, placed in

the land near to beach shores, are used to monitor the wave characteristics in an almost continuous operation. The main characteristics used to obtain the bathymetry from wave recordings are the wave breaking and dissipation patterns (van Dongeren et al. [2008]) and the already mentioned bathymetry inversion. A lot of different algorithms have been created based on this approach. Some of them have tried to extend the method using nonlinear extensions of the dispersion equation (Catálan and Haller [2008], Flampouris et al. [2011]), but the main approach consists in using the Airy dispersion relation. In order to obtain frequency and wavenumber from images, there are also different methods: 3D-FFT (Trizna [2001]), Fourier and dispersion relation fittings (Senet et al. [2008]), and spectral cross-correlations (Holman et al. [2013]). There is a more exhaustive description in Holman and Haller [2013]. However, the errors from all this methods are too big to be useful. Therefore, the measurement of bathymetry by means of remote sensing is still an open field. One of the most recent and popular bathymetry inversion algorithms is cBathy (Holman et al. [2013], Rutten et al. [2017]). Nowadays, the progress in the field seems oriented to either improving cBathy or finding new inversion algorithms. However, to test such inversion algorithms, it is important to use a set of videos with different wave and bathymetry conditions. In order to study the performance of bathymetry inversion algorithms, they must be performed over cases with known depth. The only way to decide freely which conditions to test is to generate synthetic bathymetries and obtain the propagation of waves over them by means of a numerical simulation. Using different bathymetries may manifest the problems these algorithms can encounter so that they and the proposed solutions can be subsequently analyzed.

The goal of this work is double. On one hand, to use two models of wave propagation and adapt them to generate videos that can be used to study and test dispersion equation based bathymetric inversion algorithms. To simulate the wave propagation over the desired bathymetries, both linear and nonlinear Boussinesq-type equations are used. On the other hand, to analyze and improve a well-known algorithm, cBathy, and compare it to a new algorithm developed by the directors of this project, uBathy (Simarro et al., manuscript in review), by means of the synthetic wave videos generated. Also, the videos and subsequent inversions conducted are used to find which is the best video spatiotemporal resolution.

2 . Wave propagation modelling

2.1 Linear model: Airy waves

The first model used to simulate the propagation of waves over the given bathymetries is based on the linear water wave propagation equations, *i.e.*, assuming waves whose amplitude is infinitesimal (or very small compared to the depths over which they propagate). In this case the superposition principle holds, meaning that all possible scenarios can be written exactly as a linear combination of monochromatic waves. The mathematical form of a monochromatic wave is

$$\eta = A\cos\left(\omega t + \phi\left(x, y\right)\right), \tag{2.1}$$

where η stands for water free surface elevation, A is the amplitude of the wave, ω refers to its angular frequency (constant), x and y represent the cross-shore and alongshore positions respectively, and $\phi\left(x, y\right)$ is the spatial phase of the wave. To completely determine the wave, the values for $\phi\left(x, y\right)$ need to be computed.

The wave-vector, $\vec{k}$, is defined as the opposite of the gradient of the spatial phase. Its modulus, the wavenumber k, is related to water depth h through the angular frequency ω, as stated by the dispersion equation (Equation 1.1). The wavenumber changes from point to point, but since frequency is constant, it depends directly upon the local depth. Therefore, as the frequency and the bathymetry are known (test inputs), the wavenumber can be computed from the transcendental dispersion equation with a standard root-finding Newton routine. Then, if the orientation of the wave-vector through the domain is also determined, the spatial phase can be obtained by means of integration. This can be solved assuming an alongshore-uniform bathymetry, *i. e.*, that depth does not change in the alongshore direction. In that case, the alongshore component of the wave-vector is constant, and since it is determined at the offshore boundary, where waves are forced, the whole wave-vector and therefore the spatial phase can be obtained (Derivation 2.1).

$$\vec{k} = -\vec{\nabla}\phi\left(x,y\right) \quad \rightarrow \quad \vec{\nabla} \times \vec{k} = \vec{0} \quad \rightarrow \quad \frac{\partial k_y}{\partial x} = \frac{\partial k_x}{\partial y}$$

$$\text{alongshore uniform} \quad \rightarrow \quad \left\{ \; 0 = \partial k_x/\partial y = \partial k_y/\partial x, 0 = \partial k_y/\partial y \; \right\} \quad \rightarrow \quad k_y = \text{cte.}$$

$$k_x\left(x\right) = \sqrt{k^2\left(h\right) - k_y^2} = \sqrt{k^2\left(x\right) - k_y^2}$$

$$\phi\left(x,y\right) = -\phi_0 - \int_0^y k_y dy - \int_0^x k_x\left(x\right) dx = -\phi_0 - k_y y - \int_0^x k_x\left(x\right) dx,$$

Here, k_x and k_y correspond to the cross-shore and the alongshore components of the wave-vector, and ϕ_0 is the reference of phase (the one present in the wave at the origin of coordinates at time zero). The alongshore component of the wave-vector k_y can be obtained at the offshore boundary of the domain, using the angle of incidence of the input wave and the wavenumber derived from the local depth.

Derivation 2.1: Determination of the spatial phase for the linear wave model, under the assumption of alongshore uniform bathymetry.

2.2 Nonlinear model: FUNWAVE

The other model used to simulate wave propagation over the given bathymetries is FUNWAVE-TVD 3.0, one of the versions of FUNWAVE, a Boussinesq model written in FORTRAN.

2.2.1 Equations

The Boussinesq equations are useful to model the evolution of waves from deep to shallow water over coasts. The mass and momentum conservation equations are reduced from a 3D problem to a 2D one by truncating the Taylor expansion of the vertical profile of velocity. Then, this expansion is averaged over depth (z-derivatives are substituted by horizontal ones, under the assumption of incompressible fluid and vertical or zero vorticity). Originally, the expansion used both a dispersion parameter and a nonlinearity

parameter, but there are also recent versions that only consider the dispersion parameter (these are referred to as "fully-nonlinear" since there is no limitation on the non-linearity of the waves). Once the average has been performed, the equations can be rewritten to use either the mean magnitude over the water column (Peregrine [1972]) or its value at a specific intermediate point of the column (Nwogu [1993]), which is the option chosen by FUNWAVE. In the case of FUNWAVE-TVD, the equations implemented are those presented by Chen [2006], adding the use of a time and space locally depending reference level, presented by Kennedy et al. [2001]. The equations compute the evolution of the free surface and the horizontal velocity (at a given fraction of local depth). This fraction is a free parameter of the equations, which is usually set by comparing the phase speed *versus* depth and the amplitude *versus* depth relations of the equations with those predicted by the Airy theory, and then tuning the parameter to minimize the error to both relations simultaneously (Simarro et al. [2013]).

Regarding the numerical scheme, FUNWAVE-TVD 3.0 uses an adaptive time step based on a third order Runge-Kutta method and the Courant–Friedrichs–Lewy (CFL) condition. Spatial derivatives are discretized using a combination of finite volume and finite difference methods. Wave breaking is modelled by either using a modification in the eddy viscosity term or by changing from Boussinesq to another set of equations that capture this effect, when conditions for the breaking are met (the second option is the used by default).

2.2.2 Domain, boundary conditions and wavemaker

FUNWAVE-TVD uses a rectangular domain with a variable amount of equispaced points that build up a rectangular mesh. Attenuation sponge layers can be added at the four boundaries, having an exponential attenuation profile and a selectable thickness s. The recommended thickness for attenuation sponges is of the order of the wavelengths present in the domain (around one or two of them). There are also other types of sponges available, but they are not used in this work for they are meant to mitigate the noise that the attenuation sponges introduce in long term simulations, and the videos generated for inversion only span a few minutes (including warm up). Additionally, periodic boundary conditions can be applied to lateral boundaries. Emerging earth barriers can be used to provide boundary conditions too (this can be set by specifying negative depths).

To propagate waves into the domain, FUNWAVE-TVD uses a forcing based on the Wei et al. [1999] two-way internal wavemaker. According to Wei et al. [1999], provided that the domain incorporates sponges to avoid reflections over the domain limits, the wavemaker has to be placed inside the domain itself. The way to force the wave movement

is to add a source term over the equations. The authors of FUNWAVE chose a thick-line source, that allows to introduce monochromatic waves of a given period, amplitude and angle of incidence to the domain. Different monochromatic terms can also be combined to mimic more complex conditions, as components of the desired spectrum of the input. The wavemaker needs to be configured to be effective in generating the desired waves. Specifically, its thickness has to be at least of the order of a wavelength fraction (around one quarter of it).

2.2.3 Modifications introduced on FUNWAVE

There were a couple of issues regarding FUNWAVE that required a modification of the source code. The first one involved the output times: FUNWAVE allows the user to define the time interval desired between outputs of the variables. However, it does not adjust the time step in order to obtain a solution at exactly those times. Instead, it sets the time step fulfilling the CFL condition over the domain and, when the time of an output has been surpassed, it prints the new state of the variables. However, the present study requires the outputs to be recorded at specific times, because they are used to compute discrete Fourier transforms (DFT) inside the cBathy inversion routine. For this reason, FUNWAVE code has been altered, forcing the time step to end in the times when outputs are requested, if the required time step is smaller than the specified by the CFL condition.

The other modification is related to the waves introduced in the domain. FUNWAVE provides an option that allows to specify the components of the input waves, as a list of monochromatic terms amplitudes and angles of incidence. However, it sets their phase to a random value internally. The source term generation routine was changed to accept also a list of user-defined phases. In this way, the cases where there is a superposition of monochromatic waves can be simulated with repeatability, in order to study the influence of some parameters.

2.2.4 Shoaling tests

To test if the FUNWAVE solver is working properly, a case with known solution has been simulated and the results have been compared to it. For shoaling waves, the easiest case is the small amplitude limit, that results in Airy waves, for which amplitude and phase speed can be directly written as functions of depth.

However, the FUNWAVE equations do not exactly produce Airy waves, even in the small amplitude limit. As explained in Subsection 2.2.1, they are tuned to produce am-

plitude and phase *versus* depth relations that resemble the most those of Airy waves, but the match is not perfect, and the results worsen in deeper waters. However, if the bathymetry used prevents waves from distorting (*e.g.* alongshore-uniform case), the results of Simarro [2013] can be used to compute exactly the relations that FUNWAVE equations produce.

Therefore, to check the performance of the model, small amplitude waves are propagated over an alongshore-uniform bathymetry. Then the amplitudes and phase speeds of the results are extracted and compared to the theoretical relations computed using the method of Simarro [2013]. The case simulated is quasi 1D, using four points in the alongshore dimension and an alongshore-uniform bathymetry. Waves of 6 s of period are used. The depths of interest are those between deep waters and shallow waters. To measure shallowness, the most common parameter is kh (k the wavenumber and h the local water depth). Shallow waters correspond to $kh \leqslant 0.5$, and deep water to $kh \geqslant 3.0$. Usually the intermediate case is considered as the interval $0.5 < kh < 3.0$. For waves of 6 s, that kh interval corresponds to 25 m - 5 m. Considering a maximum amplitude over depth ratio of 1/1000, the amplitude for the wave is set to 5 mm. The cross-shore profile is built using a limiting mild-slope condition, to ensure the mild-slope assumption behind Boussinesq equations.

$$\left| \frac{1}{h} \frac{dh}{dx} \right| \leqslant C$$

The values considered for C are: 0.02 m^{-1}, 0.005 m^{-1} and 0.002 m^{-1}. They define the bathymetry (Figure 2.1).

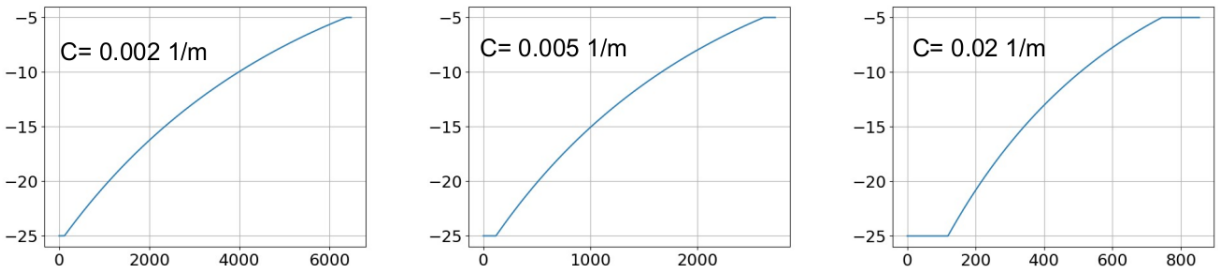


Figure 2.1: Bathymetries used for the shoaling tests, generated using $C = 0.02$ m^{-1}, 0.005 m^{-1} and 0.002 m^{-1}, respectively. The horizontal axis depicts cross-shore position in meters, and the vertical axis the bed elevation in meters. Notice that there are constant depth regions at the borders, which are used to place the wavemaker and the attenuation sponges.

To ensure that the results are correct, every case is simulated with diminishing grid size (2.0 m, 1.0 m, 0.5 m, 0.25 m). The numerical results of big grid sizes are not expected to resemble FUNWAVE theoretical relations, due to the lack of resolution in the grid size

used. For finer grids, the results should converge to the theoretical values.

However, the results of these simulations present a strong noise presence, caused by reflections in the domain limits. In order to extract the phase speed and amplitude *versus* depth from this noisy data, the output of the simulations must be processed.

The processing consists in the following steps: i) tracking the crests of the wave (finding local maxima) to get their amplitudes, positions and depths beneath for each sampling time; ii) subtract successive crest positions to get the phase speed of crests; iii) perform a time average of these magnitudes (amplitude, phase speed, depth) using a sliding window; iv) divide the range of depths into small intervals, and classify data depending on which interval falls the crest averaged depth (this is, assign each amplitude and phase speed to the interval that contains the associated depth); v) average through depth intervals using a sliding window: for each interval, take all the data that is within it and the nth nearest intervals, then compute the mean and the standard deviation of the data they contain (thus each interval gets a mean amplitude and a mean phase speed, and a confidence interval for each value); vi) use the central depth of each interval as the corresponding to its mean amplitude and phase speed. In this way, an experimental amplitude *versus* depth and phase speed *versus* depth are obtained. This processing allows to obtain results clear enough to be compared to the theoretical relations (Figures 2.2 and 2.3).

According to the figures, phase speeds stabilize for grid sizes smaller 1 m and amplitudes converge to the theoretical value for grid sizes equal or smaller than 0.5 m. Therefore, the recommended grid size is 0.5 m or below. However, this might result in too large simulation times. Considering that the amplitude is not needed for the bathymetric inversion (only the phase speed is required), grid sizes of 1 m are preferred.

2.3 Study cases and Model setup

A summary of the characteristics of the study cases can be found in Tables 2.1 and 2.2 (here ppm stands for pixels per metre). These characteristics are described in the following subsections.

2.3.1 Bathymetries

The depths of interest are of the order of meters, ranging from about 0.25 m - 0.5 m up to 8 m - 12 m. To consider a realistic case, the cross-shore bathymetry from Yu and Slinn

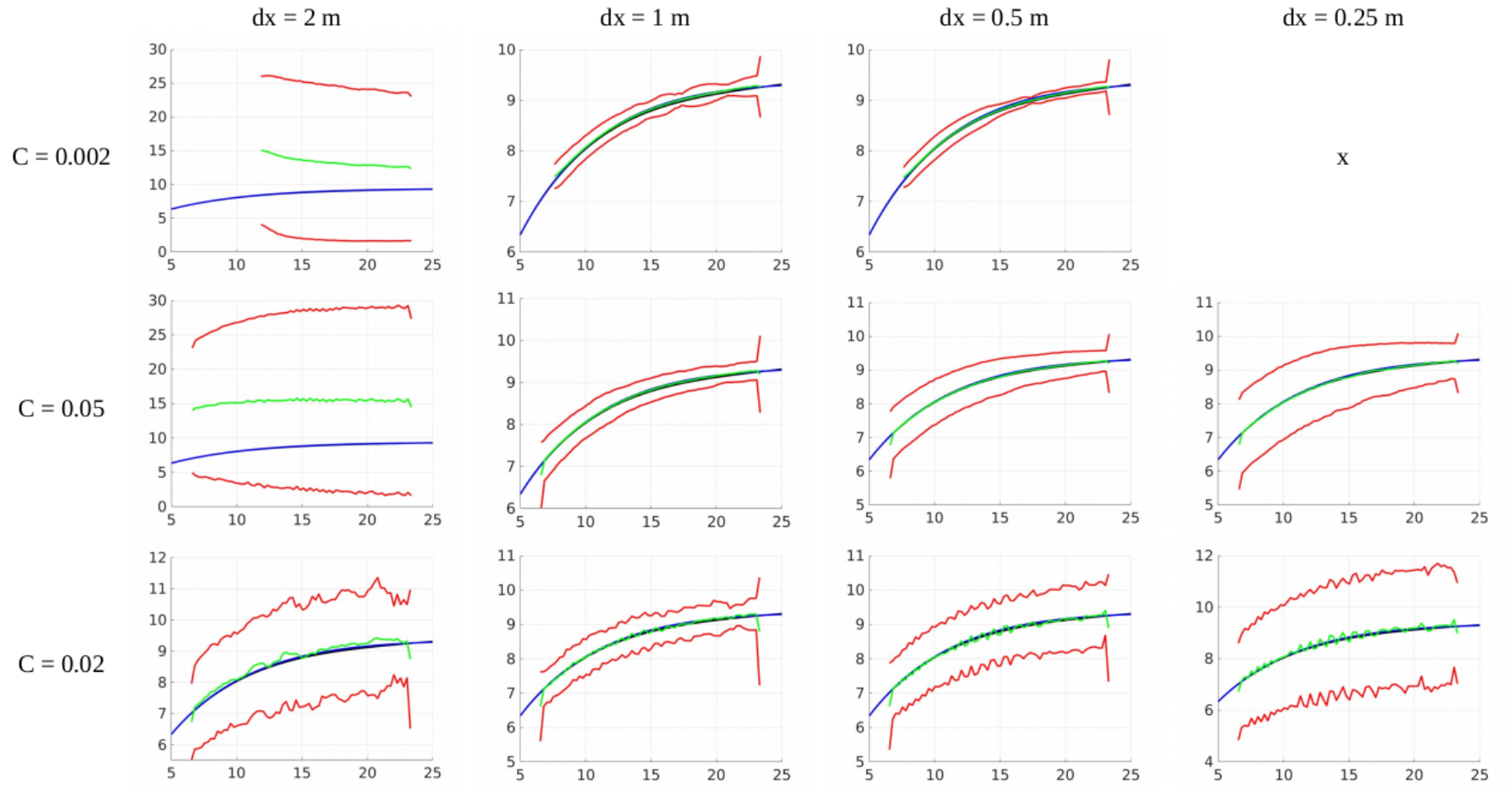


Figure 2.2: Phase speed relations for the simulated cases. The horizontal axis corresponds to depth in meters, the vertical axis shows phase speed in meters per second. The experimental relation is shown in green, its confidence intervals (one standard deviation) are plotted in red, the FUNWAVE theoretical values are in black, and the Airy ones in blue.

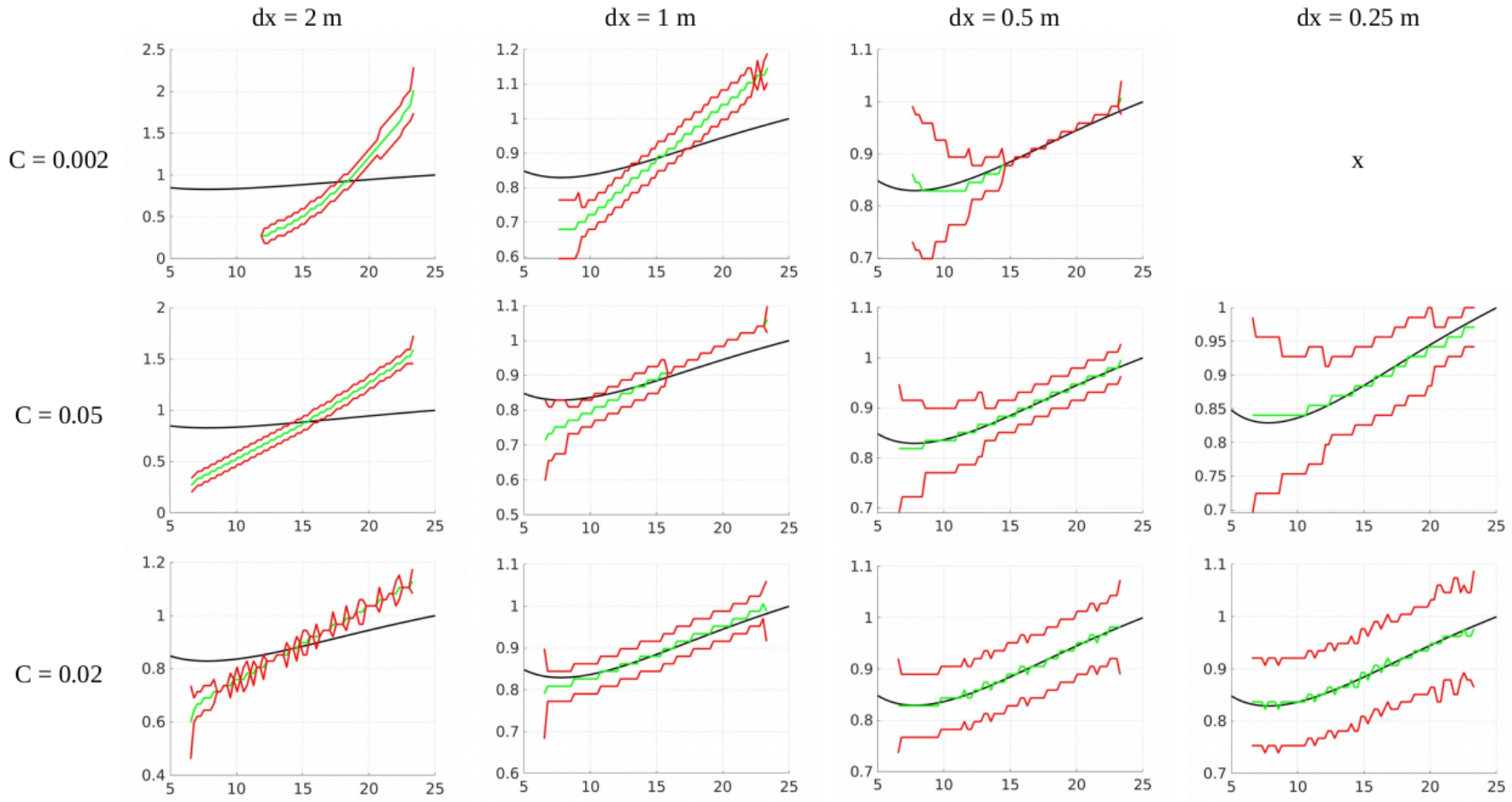


Figure 2.3: Amplitude relations for the simulated cases. The horizontal axis corresponds to depth in meters, the vertical axis shows relative amplitudes. The theoretical amplitudes have been normalized using their maximum value. The experimental results have been scaled by the scale factor that minimizes the square error between the experimental values and the FUNWAVE theoretical values. The experimental relation is shown in green, its confidence intervals (one standard deviation) are plotted in red, the FUNWAVE theoretical values are in black, and the Airy ones in blue.

Linear simulated cases	
- Bathymetry:	baseline
- Wave conditions:	monochromatic, polychromatic
- Spatial resolution:	0.10 ppm, 0.25 ppm, 0.50 ppm, 1.00 ppm
- Temporal resolution:	4.00 Hz, 2.00 Hz, 1.00 Hz, 0.50 Hz
- Amplitude multiplier:	(does not apply)

Table 2.1: Characteristics of the linear simulated cases, where ppm means pixels per metre.

FUNWAVE simulated cases	
- Bathymetry:	baseline, structures
- Wave conditions:	monochromatic, polychromatic
- Spatial resolution:	0.25 ppm
- Temporal resolution:	2.00 Hz
- Amplitude multiplier:	x1.0, x0.4, x0.1

Table 2.2: Characteristics of the FUNWAVE simulated cases, where ppm means pixels per metre.

[2003] is used as an alongshore-uniform baseline.

$$h_0(x) = \left(a_1 - \frac{a_1}{\gamma_1}\right) \tanh\left(\frac{b_1 x}{a_1}\right) + \frac{b_1 x}{\gamma_1} - a_2 \exp\left(-5\left(\frac{x - x_c}{x_c}\right)^2\right). \tag{2.2}$$

The values for the parameters are chosen as: $a_1 = 3.0$ m, $\gamma_1 = 5.55$, $b_1 = 0.09$, $a_2 = 1.5$ m, $x_c = 80.0$ m. The resulting profile can be seen in Figure 2.4.

An interval with constant depth needs to be placed before the seamost edge to allocate the wavemaker for the FUNWAVE model, so the depth is clamped at 8 m, fixing the cross-shore range for inversion tests between 300 m and 400 m. To simulate the presence of a crescentic bar, alongshore-wise undulations (Equation 2.4) and a bump (Equation 2.5) are added to the baseline bathymetry.

$$h(x,y) = m(x,y)\, h_0(x) + h_1(x,y) \tag{2.3}$$

$$m(x,y) = 1 + A_{yw} \exp\left(-5\,\frac{(x - x_{c,yw})^2}{x_{wid,yw}^2}\right)\left(-1 + \cos\left(\frac{2\pi y}{L_{y,yw}}\right)\right) \tag{2.4}$$

$$h_1(x,y) = -A_{bump} \exp\left(-\frac{(x - x_{bump})^2 + (y - y_{bump})^2}{W_{bump}^2}\right) \tag{2.5}$$

The values for the parameters needed by those equations are chosen as $A_{yw} = 0.15$,

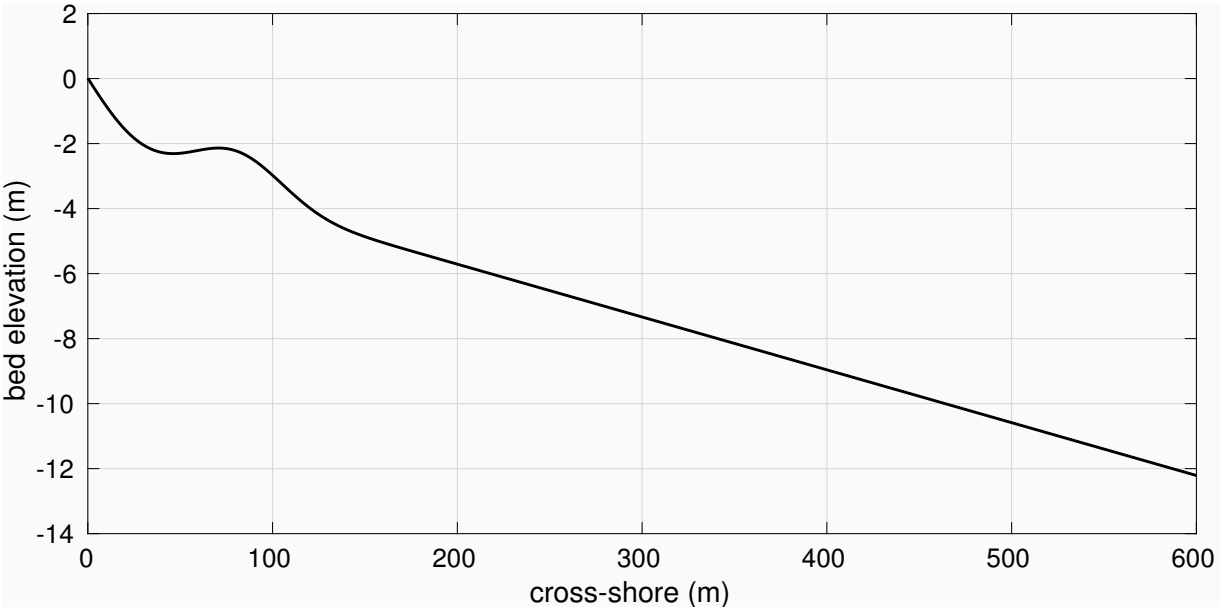


Figure 2.4: Bed elevation of the baseline considered for the test bathymetries (clipping not shown).

$L_{y,yw} = 100.0$ m, $x_{c,yw} = 60.0$ m, $x_{wid,yw} = 40.0$ m, $A_{bump} = 0.5$ m, $x_{bump} = 40.0$ m, $y_{bump} = 125.0$ m, $W_{bump} = 15.0$ m.

In order to contain these depth variations, the alongshore size is set to 200 m. The bathymetries considered can be either the alongshore-uniform baseline (hereinafter baseline case) or that same baseline combined with the undulations and the bump as described above (hereinafter structures case). The linear simulations used require an alongshore-uniform bathymetry so that for linear simulated cases only the baseline bathymetry case is used. The bathymetries used are depicted in Figure 2.5.

2.3.2 Wave conditions

Regarding the offshore wave conditions, three cases that span the range of usual frequencies, amplitudes and orientations are selected (Table 2.3). Their phase at time zero is also set (this is, the phase of their spectral representation), in order to make them repeatable and reproducible.

It is interesting to consider also another wave condition consisting in the sum of these three monochromatic components, because natural sea conditions are typically of this kind. This combined case will be referred to as "polychromatic". In order to restrict the number of test cases, only one of the monochromatic waves listed will be presented here, the one with ID 1 in Table 2.3. This wave condition will be referred to as "monochromatic".

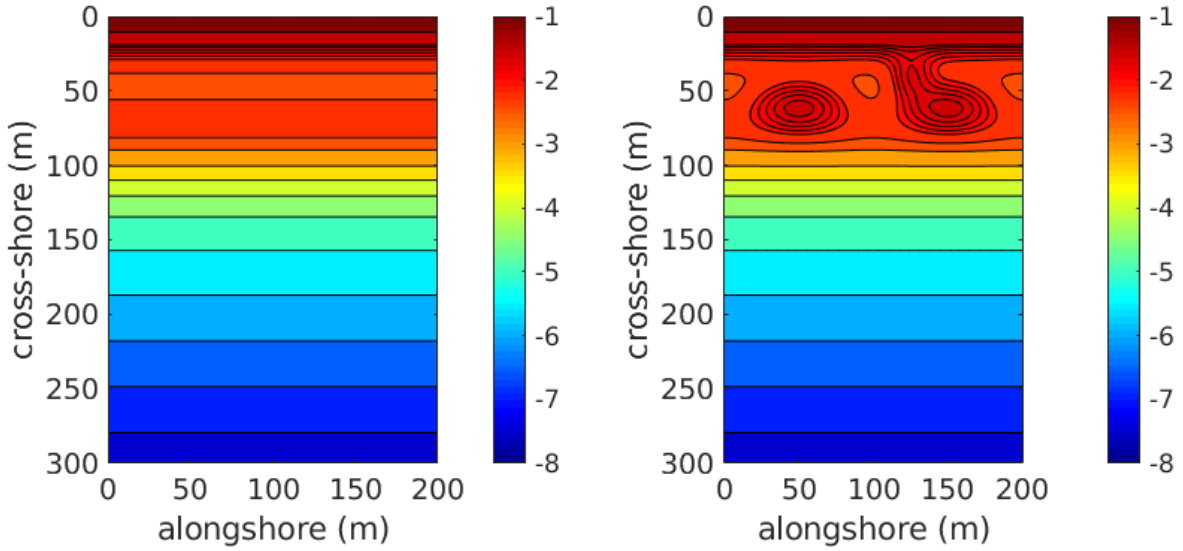


Figure 2.5: Bathymetry for the baseline case (left) and the structures case (right). The colour code corresponds to bed elevation, in meters.

ID	T [s]	f [Hz]	A [m]	θ [º]	ϕ_0 [º]
1	7.945	0.125	0.250	-16.6	39.0
2	12.000	0.083	0.150	+0.0	0.0
3	5.022	0.199	0.050	+26.1	108.7

Table 2.3: Characteristic of the monochromatic waves used. T stands for wave period, f is its frequency, A symbolizes its amplitude, θ represents its orientation and ϕ_0 its phase at time zero. The values shown correspond to the properties of the waves where they enter the domain (for the linear model, the offshore boundary, for the FUNWAVE model, the wavemaker).

The real wave conditions one can observe in a beach span multiple amplitudes. To test the limit of small amplitude and to study the dependence of nonlinear effects with amplitude, FUNWAVE simulations are also conducted with smaller amplitudes, specifically, with the amplitudes described in Table 2.3 multiplied by $\times 0.4$ and $\times 0.1$, together with the $\times 1.0$ case. This only affects FUNWAVE simulations, since linear equations are based on infinitesimal amplitude waves (in this case, the amplitudes from Table 2.3 define only the relative amplitudes for the polychromatic cases).

It is important to note that the characteristics of the waves are referred to their value at the seamost side of the domain. This is, waves are meant to come from open sea, and in the simulations their state is defined at the point they enter the domain.

2.3.3 Numerical implementation

Simulation results are obtained with a grid size of 1 m both in cross-shore and alongshore (spatial resolution of 1.0 ppm), and their output is printed each 0.25 s (sampling frequency of 4.00 Hz). However, it is important to see how the inversion performance changes with the spatial and temporal resolutions considered. To do so, the resulting images are decimated to consider also cases with 0.5 ppm, 0.25 ppm and 0.1 ppm (where ppm stants for pixels per metre, so they corresponding to 2 m, 4 m and 10 m distances between consecutive pixel centres), and sampling frequencies of 2.00 Hz, 1.00 Hz and 0.50 Hz (corresponding to sampling periods of 0.5 s, 1.00 s and 2.00 s, respectively). To decouple the dependence of results with spatiotemporal resolution from their dependence with amplitude, the spatiotemporal variations are only studied in the linear model simulations. For FUNWAVE simulated cases, the spatial resolution is set to 0.25 ppm and the sampling frequency is fixed at 2 Hz.

Finally, the video temporal span is set to 150 s, at the request of uBathy developers. To check this does not affect cBathy, consider that the windowing produced by 150 s of observation transforms the spectra of the monochromatic components, which are Dirac deltas, into the Fourier transform of a square pulse (hereinafter, a sinc). The bandwidth of a sinc can be regarded as the bandwidth of its main lobe, which is about $1/T_{obs}$, in this case, about $1/150 = 0.0067$ Hz. According to Rayleigh criterion, to resolve them, they need to be spaced more than this width. The spacing between components with ID 1 and ID 2 is 0.125 Hz - 0.083 Hz = 0.042 Hz $>$ 0.0067 Hz, the spacing between components with ID 1 and ID 3 is 0.199 Hz - 0.125 Hz = 0.074 Hz $>$ 0.0067 Hz, and the spacing between components with ID 2 and ID 3 is 0.199 Hz - 0.083 Hz = 0.116 Hz $>$ 0.0067 Hz. Therefore, a video duration of 150 s is enough for cBathy.

2.3.4 FUNWAVE model setup

The FUNWAVE simulation domain used consists in a rectangle of 672 m in cross-shore and 480 m in alongshore (Figure 2.6). The area used for bathymetry extraction (hereinafter called the video zone) measures 300 m in cross-shore and 200 m in alongshore. However, some extra space is needed for the seawards boundary (where the wavemaker is placed), the lateral boundaries (which should emulate the continuity of the coast) and the coastal boundary (where most of the dissipation or reflection of waves occur).

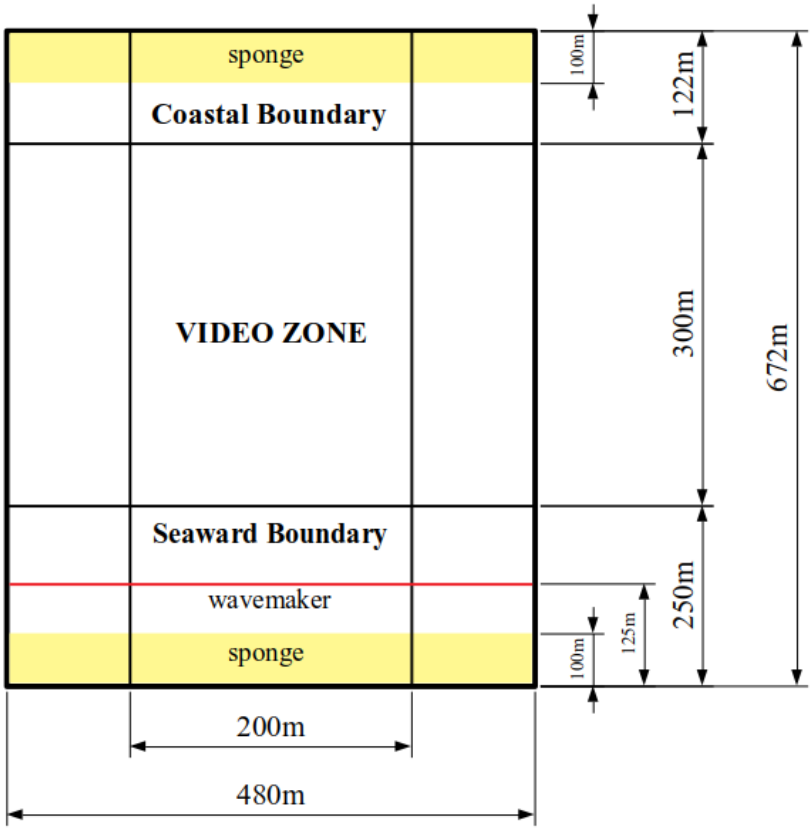


Figure 2.6: Diagram showing the regions of the domain and their size.

1. **Seaward boundary**. At least one wavelength width sponge is needed. The wavemaker needs to span another quarter of wavelength. At 8 m of depth, the wavelengths corresponding to the frequencies used are around 100 m, 65 m and 35 m. The worst case is 100 m, which means extra 125 m, although 150 m are used to left some space between the sponge and the wavemeker. However, in order to ensure that the wavemaker influence does not show in the video zone, 100 m are added between them, resulting in a total width of 250 m. Since the wavemaker needs to operate at a constant depth, the baseline bathymetry was clamped at the level it presents during the 100 m between the video zone limit and the wavemaker start (8 m, as described in Subsection 2.3.1).

2. **Lateral boundaries**. In order to avoid the effects from the lateral borders, some space at the sides of the video zone is left. Two options have been tested for the lateral boundary conditions: attenuation sponges with a reflective wall, and periodic boundary conditions. Since the attenuation sponges require at least one wavelength width (100 m) to ensure enough attenuation is provided, 140 m are added to each side. To apply the periodic boundary conditions, the alongshore wavelength must be a submultiple of the alongshore domain size. Thereby, the orientation of the incoming offshore waves has been computed to match the 480 m alongshore length. Attenuation sponges have been proved inadequate for this case, because the laterals attenuation distorts wavefronts, just as diffraction through a hole (Figure 2.7). On the contrary, periodic boundary conditions with no attenuation seem to work properly.

3. **Coastal boundary**. The first option that has been used to define boundary conditions in this side of the domain consists in keeping the natural evolution of the bathymetry described, resulting in a planar beach emerged from water. However, this setup produces a strong reflection (Figure 2.8). In order to mitigate it, a sponge is added after the video zone, with a thickness, s, of 100 m or 200 m. The bathymetry is changed near the coast, by bringing it to a constant value before the depth becomes negative. A logarithmic splice of the form "$A \log\left(10^{Ax} + 10^{C}\right) - C + L$" (modelling a two slope joint) is used to this effect. The splice starts 15 m before the video zone ends, and gradually changes its elevation gradient until the constant value of depth desired, L, is reached. The tested values for the clipping depths are 0.3 m and 0.03 m. Results are pretty similar in the four cases (*i. e.* changing s and L) (see Figure 2.9), and therefore the computationally cheapest case ($L = 0.3$ m, $s = 100$ m) is implemented for the inversion tests. However, reflections are not completely removed, though they become smaller. This has motivated the development of new strategies to cope with reflections over the coastline in the inversion algorithms.

The bathymetry inversion algorithms need 150 s of video, but 200 s of simulation are added for the simulated system to warm up. Therefore, the total simulation time is 350 s. All the simulations used to test the boundaries have been conducted using 1 m of grid size, both in cross-shore and alongshore directions. This has kept computation times into a reasonable order of magnitude. To check that this is enough, a couple of simulations using 0.5 m of grid size have been conducted. Visual inspection confirms that the results are similar enough for the purposes of this work.

To speed up simulation times, the model is parallelized. The way FUNWAVE incorporates it is by subdividing the computation domain into smaller ones. To do so, it needs the number of points in the subdivided direction to be a multiple of the number of cores

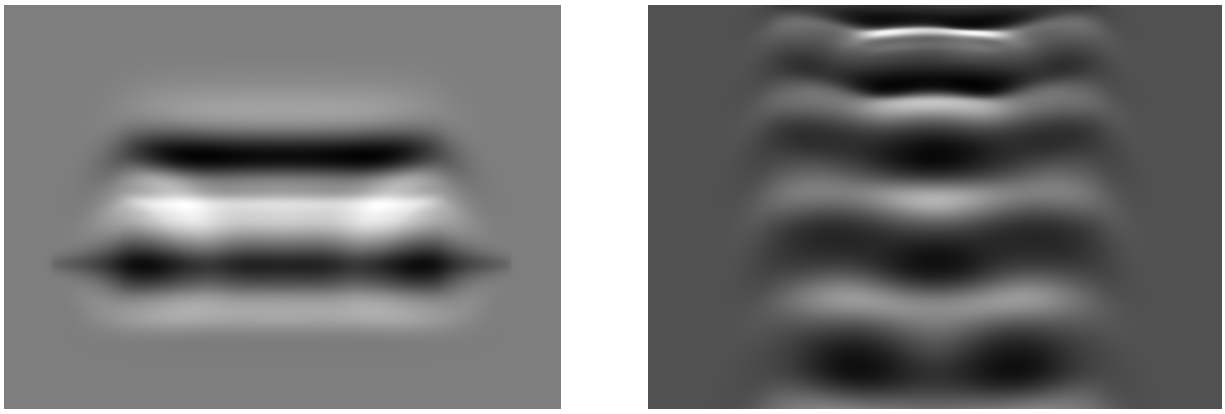


Figure 2.7: Free surface maps showing the effect of sponges and reflective walls as lateral boundary conditions. The left image shows some circular waves that appear due to the attenuation of the main plane wave wavefronts at the lateral boundaries. The right image shows a snap for a later time instant. Notice how the wavefronts are not straight lines as expected, due to the interaction with the circular waves. The images are build as explained in Section 2.4. The lateral and the offshore margins of the domain are depicted too, beside the video zone.

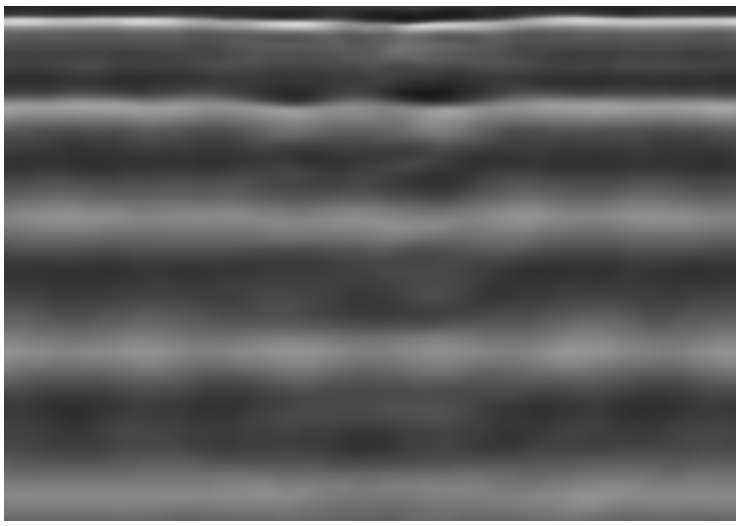


Figure 2.8: Free surface maps showing the reflections caused by the emerged land coastal boundary conditions. The contrast between crests and valleys is quite small due to reflections. The form of the reflected wave is barely visible, but it may be observed if the image is looked from some distance. The lateral margins are depicted too, beside the video zone.

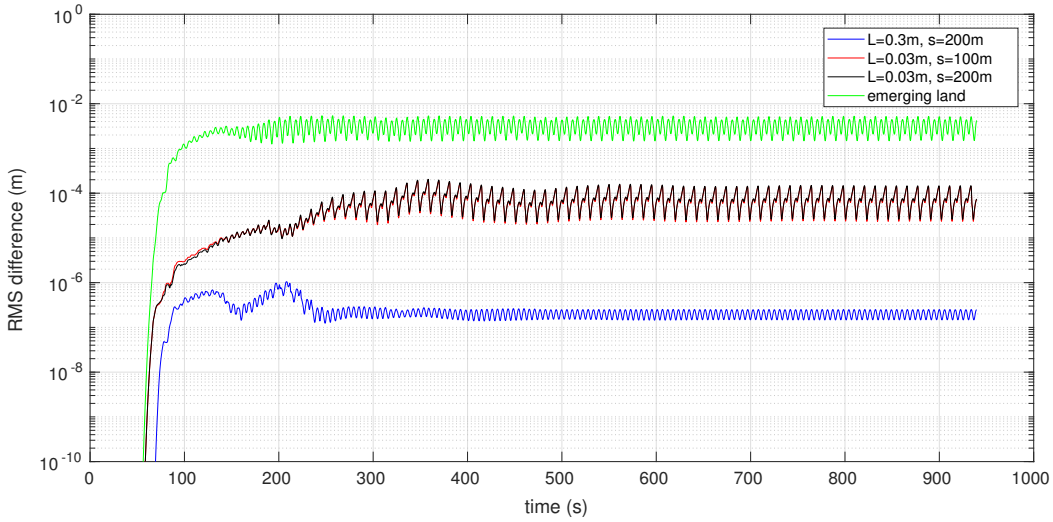


Figure 2.9: Temporal evolution of the RMS difference in free surface height among the domain, compared to the 0.3 m clipped 100 m thickness sponge case. The horizontal axis corresponds to simulation time in seconds, and the vertical axis is the RMS difference between pixel's free surface elevation. The case of emerging coast is also displayed.

used. Since systems used have up to 24 cores, it was ensured that both sides of the domain have a number of points multiple of 24 (this leads to a slightly longer coastal zone after the video zone, as can be seen in Figure 2.6).

To implement all the conditions for each simulation while allowing to change these parameters in a fast way, a routine combining Bash scripting and Python have been developed. It gets the information from the input file (where the positions of the wavemaker, the size of domain parts, the thickness of the sponges, the input wave spectrum and the clipping depth are described), computes the inputs necessary (changing limits to match a number of grid points suitable for parallellization and computing the bathymetry according to the analytic expressions and the clippings), and feeds them to FUNWAVE.

2.4 Results

Finally, free surface height resulting from simulations are used to build images, by means of a linear transformation to 128 grayscale levels. The following pages contain some examples of the images built. Figure 2.10 shows the three monochromatic waves used in this work, alone and combined. Notice how the crests of the waves get closer near the coast, as the depth becomes smaller. In some pictures it is also possible to see how the crests become more intense, indicating a bigger amplitude due to shoaling. The cases that use

the bathymetry with structures show the refraction of the incoming waves when they pass over the undulations and the bump, resulting in bent wavefronts. The different amplitude cases considered are depicted in Figure 2.11. The difference between crests and valleys is more definite for bigger amplitudes. Finally, Figure 2.12 depicts the different spatial resolutions used for inversion.

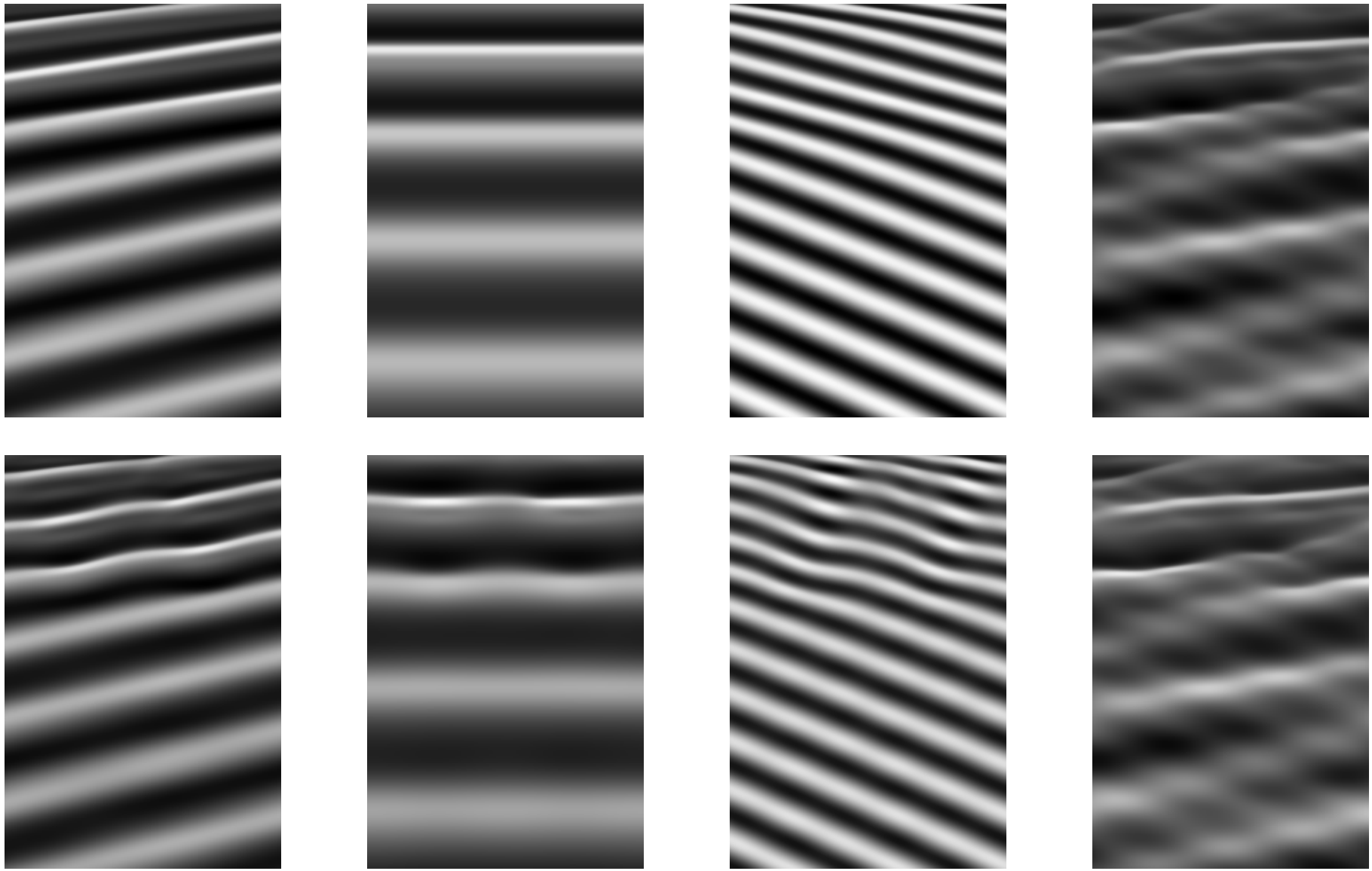


Figure 2.10: Images resulting from the simulations. The top row corresponds to cases with the baseline bathymetry, and the bottom row to cases with a bathymetry with structures. The wave of each column correspond to ID 1, 2, 3, and the combination of them three. All the simulations have been done with FUNWAVE. The resolution is 1.00 ppm, and the amplitude factor is x1.0. The coast is up.

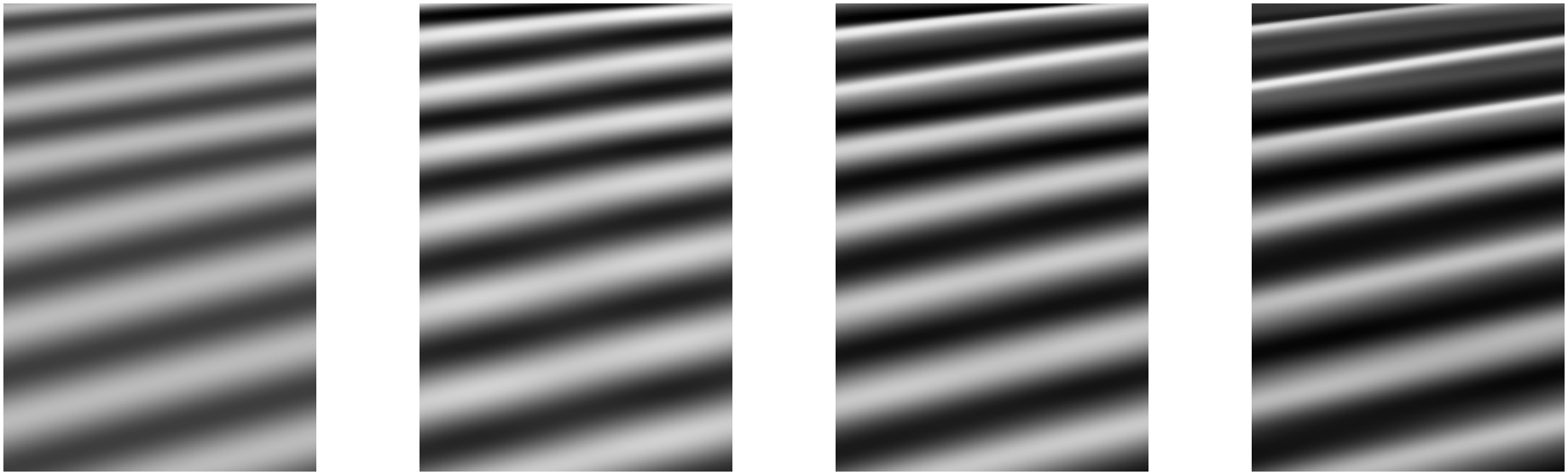


Figure 2.11: Resulting images for waves with ID 1, for different amplitude factors. From left to right, linear model simulated, FUNWAVE simulated with amplitude x0.1, FUNWAVE simulated with amplitude x0.4 and FUNWAVE simulated with amplitude x1.0. The spatial resolution is 1.00 ppm. Baseline bathymetry has been used. The coast is up.

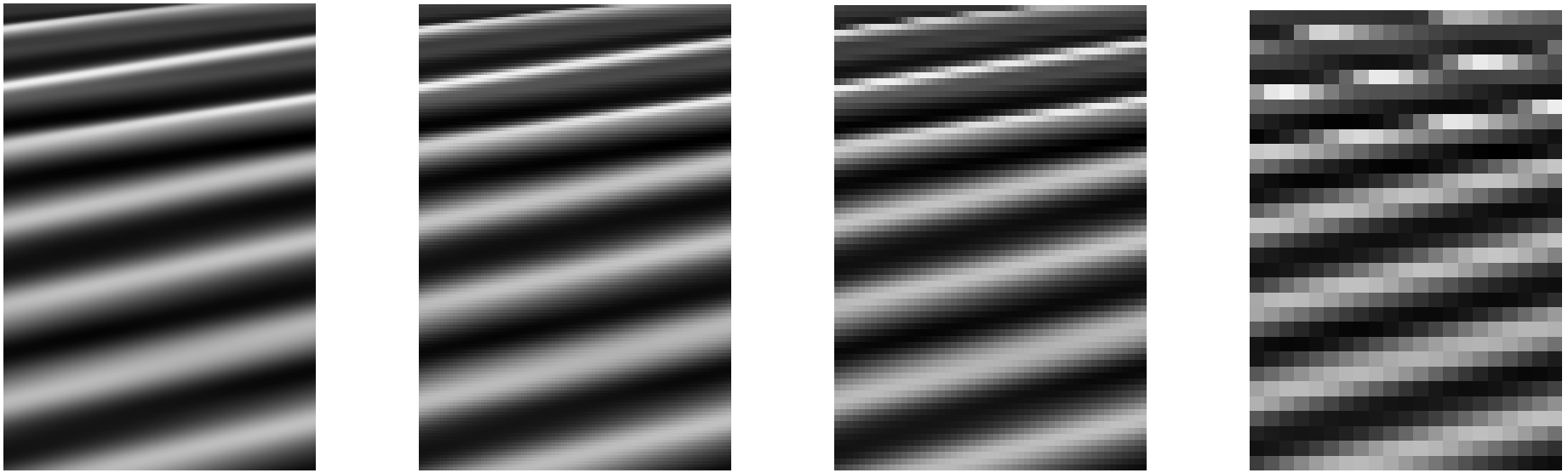


Figure 2.12: Resulting images for waves with ID 1, with different spatial resolutions. Respectively, these are 1.0 ppm, 0.5 ppm, 0.25 ppm and 0.10 ppm. They have been simulated with the FUNWAVE model (amplitude x1.0, baseline bathymetry). The coast is up.

3 . Inversion Algorithms

3.1 cBathy

CBathy is a popular bathymetric inversion algorithm that uses Airy dispersion equation (Equation 1.1) to guess the local depth of water. According to its authors (Holman et al. [2013]), cBathy consists in a three stage routine that i) guesses waves frequencies and estimates wavenumbers for each frequency; ii) combines different frequency information to return a batch estimate of depth; iii) fuses batch estimates of depth along time using a Kalman filter to provide better estimates. This work focus in the first two stages of the algorithm.

For the moment, the algorithms are only officially implemented in Matlab. The version 1.2 of the code provided in the Coastal Imaging Research Network repository has been modified and used.

3.1.1 Methodology

In order to apply the dispersion equation, cBathy first needs to estimate wave frequencies and the spatial distribution of the wavenumbers through the domain of inversion.

The input for the inversion is a video of wave amplitude evolution over time in a 2D domain. The wave amplitude can be modelled as a sum of monochromatic terms (described in Equation 2.1).

$$s(\vec{r},t) = \sum_c s_c(\vec{r},t) = A_c(\vec{r})\ \cos\left(2\pi f_c t + \phi_c(\vec{r})\right) \tag{3.1}$$

where $s(\vec{r},t)$ is the measured signal, c is an index that lists the monochromatic components present in the signal, $s_c(\vec{r},t)$ are the monochromatic components, and f_c are the linear frequencies of each component. CBathy takes each pixel temporal series and applies a Fourier transform to get its data in the frequential domain. In time, all the waves that are passing through a point are mixed, but in frequency they become decoupled.

$$\mathcal{F}\left\{s(\vec{r},t)\right\}(f) = \sum_c \frac{T}{2}\ A_c(\vec{r})\ \left(e^{j\phi_c(\vec{r})}\ \mathrm{sinc}(f-f_c) + e^{-j\phi_c(\vec{r})}\ \mathrm{sinc}(f+f_c)\right)$$

where $\mathcal{F}$ represents the Fourier transform, j is the imaginary unit, and "$\mathrm{sinc}(\pi T f) = T\sin(\pi T f)/\pi T f$" represents the Fourier transform of a square pulse of duration T. If the

sampling frequency is high enough to avoid aliasing, the terms that contain a $\mathrm{sinc}(f + f_c)$ can be neglected.

$$\mathcal{F}\left\{s(\vec{r},t)\right\}(f) \simeq \sum_c \frac{T}{2} A_c(\vec{r})\, e^{j\phi_c(\vec{r})}\, \mathrm{sinc}(f - f_c)$$

Also, if the monochromatic terms frequencies are far enough from each other (see last part of Subsection 2.3.3), the different terms do not mask each other, *i. e.*, the value of the pixel spectrum around the frequency of a specific monochromatic term f_{c_1} is $\frac{T}{2}A_c(\vec{r}) \cdot e^{j\phi_{c_1}(\vec{r})}$.

Therefore, if the frequencies of the monochromatic components can be identified, their spatial phases can be retrieved just by reading the phase of the corresponding coefficients of the pixels spectra. Actually, cBathy uses directly the normalized Fourier transform of each pixel signal, to work only with the spectral phase.

$$G_m(f) = \frac{\mathcal{F}\left\{s(\vec{r}_m,t)\right\}(f)}{\left|\mathcal{F}\left\{s(\vec{r}_m,t)\right\}(f)\right|}$$

where m is an index that lists the pixels, $\vec{r}_m$ are the positions of the pixels, and G_m are the phases of their spectra.

However, in nature data is highly corrupted with noise, so that it is difficult to distinguish the presence of different waves directly over a pixel spectrum, and to extract clean spatial phases from it. In order to overcome this problem, cBathy uses two different techniques. The first one is to use spectral correlation. Specifically, since the amplitudes of all spectra have been neglected, it performs a phase correlation. It consists in computing the complex phase lags between two pixels spectra ($\overline{G_m(f)}G_n(f)$). For each frequency, this defines a matrix of spectral correlation.

$$C_{m,n}(f) = \overline{G_m(f)}G_n(f)$$

If the signal to noise ratio (SNR) of the videos is high enough, the eigenvector with the biggest eigenvalue of $C_{m,n}$ can be assumed to be a better approximation to the spatial phase than the raw coefficients of $G_m(f)$, since it gets rid of part of the noise (the part that is orthogonal to it). However cBathy does not compute the eigenvectors of $C_{m,n}(f)$ directly, but uses a second technique to get even a better approximation. It averages $C_{m,n}(f)$ along frequency, using a set of user defined bands. For each band, it takes all the $C_{m,n}(f)$ that correspond to frequencies inside that band, and computes their mean frequential value. CBathy authors call it cross-spectral matrix (CSM).

$$CSM_{m,n,b} = \frac{1}{L} \sum_{f_l \in band_b} C_{m,n}(f_l) \tag{3.2}$$

where l is an index that lists all the frequencies inside a band b (the f_l's), and L is the number of frequencies inside that band. If a band contains only noise, the elements of $C_{m,n}$ will oscillate quite fast with frequency. In contrast, if the band contains a clear signal (*i. e.*, there is a wave with that frequency), the elements of $C_{m,n}$ will present an almost constant behaviour, since the corresponding sinc has a constant phase. If the band contains a signal but the level of noise is comparable to the wave signal or stronger than it (this is, the SNR is not that good), the phase will no longer present an almost constant behaviour, but it will be a sum of these two terms, one constant and one oscillating fast. Therefore, when summing the values of different frequencies to compute the average, the noise terms will add up in an incoherent way (provided that they oscillate rapidly with frequency) while the signal terms will add coherently (due to its constant behaviour with frequency). Then, $CSM_{m,n,b}$ will present a greater SNR than the different $C_{l,m}(f)$ used to compute it. However, this introduces two problems: the resolution to determine monochromatic components is deteriorated (by a factor L), and it adds the requirement to define the set of averaging bands to the user (the cBathy authors recommend to use about 20 - 40 of them). But, since $CSM_{m,n,b}$ presents lowers levels of noise in relation to the levels of signal, using the eigenvector associated to its greatest eigenvalue provides even a better estimate to the spatial phase for waves with a frequency inside the band b, and this is the procedure that cBathy follows. This eigenvector is hereafter called phase array ($v[m]$).

Moreover, the computation of $CSM_{m,n,b}$ provides also a way to find which bands are more likely to contain a monochromatic wave. As described above, the bands containing only noise will hold elements of $CSM_{m,n,b}$ of smaller amplitude than those bands that contain a signal [1]. In order to get rid of outliers and capture the general tendency for that band, the mean amplitude over the whole set of $CSMm, n, b$ for band b is computed. The authors of cBathy call it squared coherence $coh2$ (hereinafter, coherence) [2].

$$coh2[b] = \frac{1}{M} \sum_{m}^{M} \sum_{n}^{M} \left| CSM_{m,n,b} \right| \tag{3.3}$$

[1]The sum of L coherent complex exponentials is $\sim L$, while the sum of L incoherent complex exponentials is $\sim \sqrt{L}$. Therefore, the average will present an amplitude close to one for bands containing a wave, and an amplitude of the order of 1 / $\sqrt{L}$ for bands containing only noise.

[2]However, the classical definition of magnitude squared coherence is defined taking the square of the spectral correlation (normalized by the correlated power spectra). That is why it is called "squared".

CBathy sorts the values of the coherence, and assumes that the bands with the biggest coherence contain a wave. Then it assumes that that each of these waves has a frequency that corresponds to the one of the center of the band, and a spatial phase that corresponds to the values of the associated phase array.

However, there is another trick involved, to find the spatial distribution of wave-vectors that correspond to that spatial phase. Instead of computing the $C_{m,n}$, the $CSM_{m,n,b}$ and the $v[m]$ of the whole domain, it does it for a small neighbourhood around each pixel (hereafter, a tile). In this way, the memory required to the computations can be easily allocated, the matrices are small enough to be easily manipulated, and the wave-vector of each monochromatic term can be assumed constant over the whole tile. The phase array presents the following mathematical form.

$$v[m] = \exp(j(-\vec{k} \cdot \vec{r}_m + \phi_0)) + \varepsilon_m \tag{3.4}$$

where ϕ_0 is the phase at the origin, and the ε_m symbolize the errors between the exact spatial phase and the phase array. cBathy tries to fit the phase array to this model (Equation 3.4), and returns the norm of the fitted wave-vector as the wavenumber corresponding to that monochromatic component, over that pixel.

Once it has gathered frequency-wavenumber pairs for all the pixels, cBathy loops again through them. For each point, it takes all the pairs estimated inside its tile and fits the dispersion equation to them, using depth as the free parameter.

3.1.2 Modifications implemented on cBathy

The cBathy code has been modified in different ways, in order to improve its performance and to adapt it to the necessities of this work [3].

1. **Fourier Transform**. The possibility to apply a window to the data before Fourier transforming it has been added to the code, to try to improve the identification of peaks in the frequency spectrum for some cases. However, after trying different windows, the rectangular one (no windowing) appears to work better than the others, at least for the cases tested. This may be related to the amount of noise present

[3] Additionally, the cBathy code was translated into Python, incorporating all these changes. At first it was observed that this version ran quite slowly. The reason was that the version of NumPy used had not optimized DFT routines. This problem was solved by adding a wrapper to the C version of FFTW. However, the results of the inversion were not as good as the Matlab ones. This issue has not been solved yet. Maybe the optimization libraries that perform the fitting in Matlab are more robust that those of SciPy in Python.

in the data (the rectangular window is the one with the smallest equivalent noise bandwidth). The option to return and use more samples of the spectrum was also implemented. Usually, when a DFT is performed, the result has as many samples as the original signal, equispaced in the frequency domain between zero and the sampling frequency. However, it is possible to evaluate the Fourier transform at other frequencies (this is normally done by zero padding the data before applying the DFT routine). Adding more samples seems beneficial for the inversion. This may happen because having more samples per band increases the averaging effect that allows the correlation and coherence to distinguish between bands containing signal or just noise. It also makes results more stable, as described in the next section. Finally, the moment when Fourier transforms are computed inside the cBathy algorithm has been changed. Originally, they are obtained at the very beginning of the routine. However, since inversion domains used are larger (they contain more points) than those of the authors, and since more samples are demanded to the spectrum of each pixel, too much memory is required. To solve this, DFTs are now computed after selecting the tile. In this way, only the the spectra of the points that are inside it needs to be stored. This means that if a point appears in more than one tile its DFT computation is repeated, but this is not a problem because computing a DFT is done in a relatively fast and cheap way.

2. **Band preselection**. Since the frequency of waves is constant over the whole domain, the bands containing a signal can be searched and found before looping over all the tiles. Thus, cBathy has been modified to give the option to compute the correlation only over the preselected bands. The main benefit obtained is that the amount of computations is reduced, not only by computing less CSM slices (from around 40 to 2 or 3 of them) but also by skipping the coherence calculation and its sorting for each tile. Band preselection results in using the same bands over the whole domain. On the one hand, this makes wave guessing more consistent, since all tiles search wavenumbers for the same frequencies (thus one can study how wavenumber estimation performance depends on tile position), and it also prevents tiles that generate poorly band estimated from searching at incorrect frequencies. On the other hand, it may make the algorithm less robust, cause if the preselected bands are not estimated correctly, or if the wave frequency lays between two bands, results worsen. This feature can be selected using a boolean variable.

3. **Band averaging**. The option of performing a weighted average in coherence computation, instead of a regular one has been included to cBathy. Weights are computed using a symmetric window that has a maximum at its centre. This feature is not very useful by itself (if bands are used to find coherence, the best results are obtaining

using equal weights), but in combination with a coherence computed continuously (explained in Subsection 3.1.4), the weighted average provides a better coherence.

4. **Wave-vector fitting weights**. CBathy uses weighted residuals for the fitting of the phase array to a monochromatic wave spatial term (Equation 3.4). This means that not all the errors between the modelled and measured phase array will be given the same importance. The weights used in this case are a product between phase array amplitudes and a function that decreases with the distance to the tile centre. Specifically they use an anisotropic normalized distance.

$$r = \sqrt{\left(\frac{x - x_c}{L_x}\right)^2 + \left(\frac{y - y_c}{L_y}\right)^2} \tag{3.5}$$

where (x_c, y_c) represent the coordinates of the tile centre, L_x the cross-shore radius of the tile, and L_y the alongshore radius of the tile. Also, they set to zero all the points that present a distance bigger than one, which implies that the points at the corners (the tile is rectangular) are not used. That means that their spectra, the correlations, and the eigenvectors of bigger matrices are computed for no reason. Alternatively, a normalization after the computation of this distance has been introduced to keep them all below one, thus using all the points of the tile to estimate the wave-vector.

5. **Wave-vector pre-estimate**. In order to feed the spatial phase fitting (Equation 3.4) a wave-vector pre-estimate is needed. CBathy guesses the cross-shore and the alongshore components of the wave-vector as follows. First, it orders the pixels using the other coordinate (when estimating k_x, it orders by y, when estimating k_y, it orders by x). It classifies the points into groups of the same size, and assumes they represent transects over the tile. Then, it computes the wavenumber component using differences in phase array angle along consecutive points, and takes its median as the wave-vector component that is present in that transect.

$$k_x = \text{median}\left(\frac{\angle v[m+1] - \angle v[m]}{x_{m+1} - x_m}\right) \tag{3.6}$$

where m indicates the index of a point in a transect, $\angle v[m]$ represents the angle of the phase at that point ($\text{atan2}(\text{Im}(v[m]), \text{Re}(v[m]))$), and the points are ordered by growing x. Finally, it takes the median of transects estimates as the initial guess to feed the optimization routine. Since the authors data come from different cameras oriented in different directions, their points do not form a rectangular grid over the domain. Therefore, they do not represent real transects, and then the grouping is arbitrary. In contrast, the videos generated in this project form a rectangular grid in

cross-shore and alongshore (simulations come from a rectangular grid, and real data would come from a single camera, too). Therefore, the wavenumber pre-estimation routine has been modified to take real transects, in order to have better estimates. To toggle between the two algorithms for transect making, a boolean variable has been added, that indicates if the data comes from a rectangular grid.

Also, notice that some extra outliers can appear due to phase wrapping: the angle of a complex number is multivalued, it may result in the same complex phase if 2π is added several times to it. Depending on the absolute phases of the eigenvector, the jumps of the angles can appear at any place. To solve this, the phase array has been normalized by the phase of the closest data point to the tile centre. In this way, angle jumps are displaced away from the tile centre. If the tile is smaller than a wavelength, angle jumps will not appear. The normalization is completely harmless, since the phase array contains the values for spatial phase differences between pixels, and these can be referred to any origin (there is one degree of freedom, in the form of a global phase). Also, remember that the phase array is an eigenvector (multiplying it by any complex number still makes it an eigenvector).

Another change to the pre-estimation routine has been to clip the return values. In the original code, when the pre-estimated wavenumber is out of the limit of deep and shallow water (in this case, defined by the minimum depth allowed in the bathymetry), cBathy returns an arbitrary value (the wavenumber corresponding to a frequency of 0.3 Hz and 3 m of depth). When the main routine receives a value out of the shallow and deep water bounds, it surrenders and skips the estimation of a wavenumber for that band at that tile. The return value for the out-of-bounds cases has been changed, now its value is clipped to the shallow and deep water limit (plus a small epsilon to ensure that they still remain inside the interval at comparison time). Thus, the wavenumber pre-estimate is moved to the nearest plausible value, and the fitting is attempted.

6. **Bound for wave-vector direction**. In the original cBathy code, there was a filter to the angle of the estimated wave-vector. If the estimated wave-vector represented a wave coming from the coast, the result was neglected and it did not provide an estimate for the wavenumber. That condition has been removed, because in the fitting routine one can not control if the resulting wave-vector will be pointing towards or from the coast (they both may be the best result for the optimization), and the resulting wavenumber, which is the interesting parameter, would be the same.

7. **Depth fitting weights**. CBathy also uses weighted residuals when fitting the

estimated pairs of frequency-wavenumber to the dispersion equation. The weights used are a combination of a function decaying with distance to tile centre (the same used in the wavenumber estimation), the coefficient of determination of the wave-vector fit (indicating how good was that estimate), the relative eigenvalue of the phase array (eigenvalue divided to the mean of the CSM eigenvalues, indicating how representative is that phase array), the inverse of depth sensitivity to errors in wavenumber, and the inverse of the wavenumber.

The depth sensitivity to errors in wavenumber can be computed as described in Derivation 3.1.

$$\Delta h \simeq \frac{dh}{dk}\Delta k \rightarrow \left|\frac{\Delta h}{h}\right| \simeq \left|\frac{k}{h}\frac{dh}{dk}\right|\left|\frac{\Delta k}{k}\right| \rightarrow \text{sens} \equiv \left|\frac{k}{h}\frac{dh}{dk}\right|$$

$$\frac{dh}{dk} = -\frac{1}{k^2}\operatorname{arctanh}\left(\frac{w^2}{gk}\right) - \frac{1}{k^2}\frac{\frac{w^2}{gk}}{1-\left(\frac{w^2}{gk}\right)^2}$$

$$\text{sens} = \left|\frac{k}{h}\frac{dh}{dk}\right| = 1 + \frac{\frac{w^2}{gk}}{1-\left(\frac{w^2}{gk}\right)^2}\frac{1}{\operatorname{arctanh}\left(\frac{w^2}{gk}\right)}$$

$$\gamma \equiv \frac{w^2}{gk} \rightarrow \text{sens} = \frac{1}{\operatorname{arctanh}(\gamma)}\left(\operatorname{arctanh}(\gamma) + \frac{\gamma}{1-\gamma^2}\right)$$

$$\left(\operatorname{arctanh}(\gamma) + \frac{\gamma}{1-\gamma^2}\right) = \frac{d}{d\gamma}\left(\gamma\ \operatorname{arctanh}(\gamma)\right)$$

$$\text{sens} = \frac{1}{\operatorname{arctanh}(\gamma)}\ \ \frac{d}{d\gamma}\left(\gamma\ \operatorname{arctanh}(\gamma)\right)$$

Derivation 3.1: Depth sensitivity to wavenumber errors, expressed as cBathy routines use it.

The original cBathy code uses this alternative expression, but it divides by $\tanh(\gamma)$ instead of $\operatorname{arctanh}(\gamma)$. However, the resulting weights are similar (in fact the one using $\tanh(\gamma)$ is more restrictive), as can be seen in Figure 3.1. Unfortunately, the authors do not left an explanation about this decision, and so the theoretical $\operatorname{arctanh}(\gamma)$ is the one used.

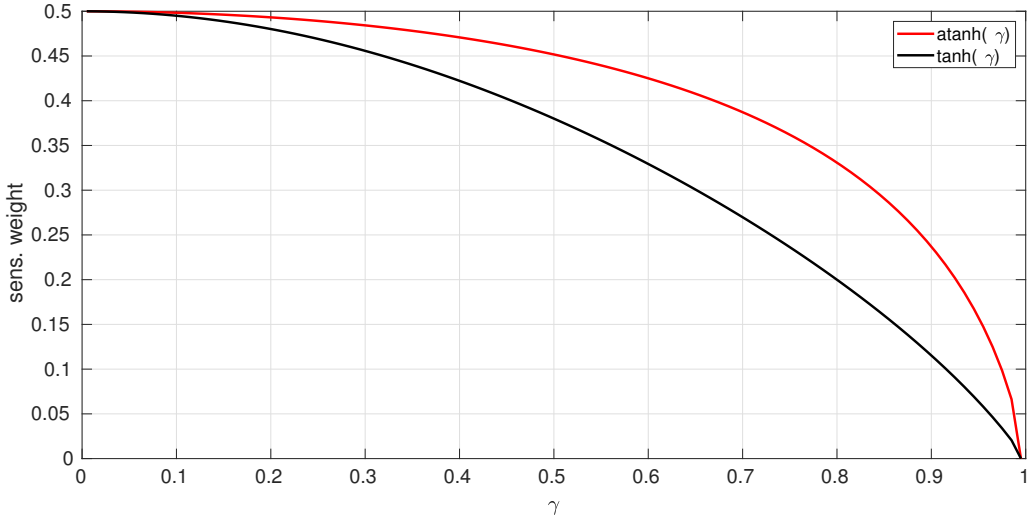


Figure 3.1: Comparison of resulting sensitivity weights (1/sens).

3.1.3 Parameter tuning

In order to run cBathy and obtain useful results, it is crucial to tune a series of parameters. The most important ones are: i) the resolution of the frequency bands used in the CSM averaging (hereafter $df B$) because it determines the resolution obtained in wave frequency estimation and also because it defines the scale of the interval over which the spectrum is averaged; ii) the resolution of the spectrum returned by the DFT (hereafter df) since once the width of bands have been defined it determines how many samples will be averaged, that need to be high enough to ensure that statistics of the averaging work properly; iii) the scaling of the tile size, because it determines how many pixels are used to compute the CSM, and therefore controls directly the quality of the phase vector obtained; iv) the level of noise relative to the signal level (hereinafter nra), because the synthetic data generated is too clean to allow a characterization of the wave frequencies by averaging the spectrum.

Instead of relying on default parameters, or to try to guess a good value for them manually, a set of routines are designed to find proper values for the important parameters automatically. However, an algorithm to tune the scaling of the tile size could not be developed, and thus this parameter is still set manually. Therefore, the proposed algorithm finds correct values for $df B$, df and nra. In order to study their effects in the determination of waves, a small tile around the centre of the domain is selected, and its associated coherence computed. Then, the effects of these three parameters on the coherence are analyzed. To do so, a proper value for each parameter is found manually, and then two of them are kept constant while the third is changed.

If dfB and df are set to proper values, and nra is set to 0, a noisy function that touches 1 almost at each band can be observed. The problem happens to be having data which is too clean: the lack of noise produces an always coherent spectrum. If noise is added to this data, the coherence plot starts to bend towards zero. The bands containing frequencies of waves present in the video keep a higher value. Adding even more noise makes the peak narrower, while it descends until all the coherence ends lying in the same level, at the noise floor (Figure 3.2). On the contrary, data coming from a real beach (the demo that cBathy authors attach to the code), presents a well defined peak even when nra is set at 0, and goes to noise floor for higher values. This reinforces the idea that data coming from real sources has an inherent amount of noise within. The question is if all the natural videos present enough levels of noise, or if their data needs to be contaminated with more noise in order to be used for bathymetric inversion.

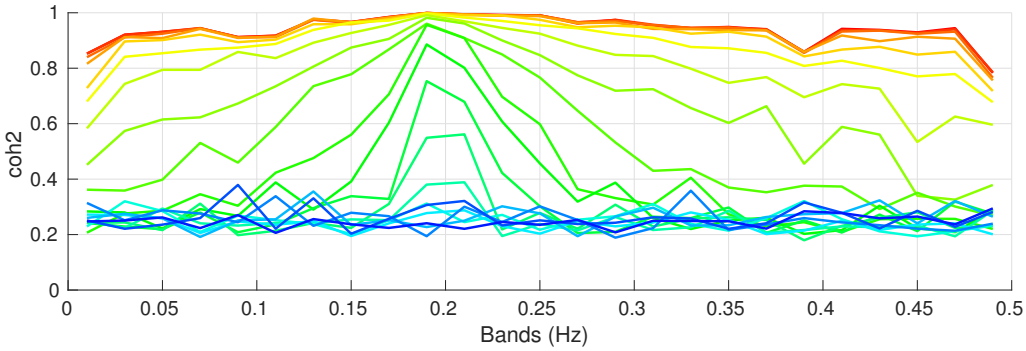


Figure 3.2: Coherence dependence with noise nra (warmer colour: less noise, colder colour: more noise).

If nra and df are kept constant within a proper value, and dfB is changed (*i. e.*, modifying in how many bands we divide the zone of interest of the spectrum), for small dfB the coherence presents fast and strong variations with frequency. As dfB becomes bigger, the oscillations become smaller in size, and curl around a noise level that becomes smaller with wider bands. Finally, the peak starts to decrease, too (Figure 3.3). in Figure 3.3.

If dfB and nra are kept fixed, and df changes from a high resolution (small df) towards a low one, the coherence function remains the same at first, but eventually it starts to tremble, and then it changes its shape completely, widening the peak and eventually ending in the case of almost constant line at coherence equal to 1 (Figure 3.4). This behaviour suggests that a high resolution value of df can be kept while dfB and nra are tuned to provide the best coherence possible, and then adjust it in order use the smallest amount of samples that keeps the coherence function almost unchanged compared to that resulting from using higher resolutions (the limit is found using a threshold on the norm

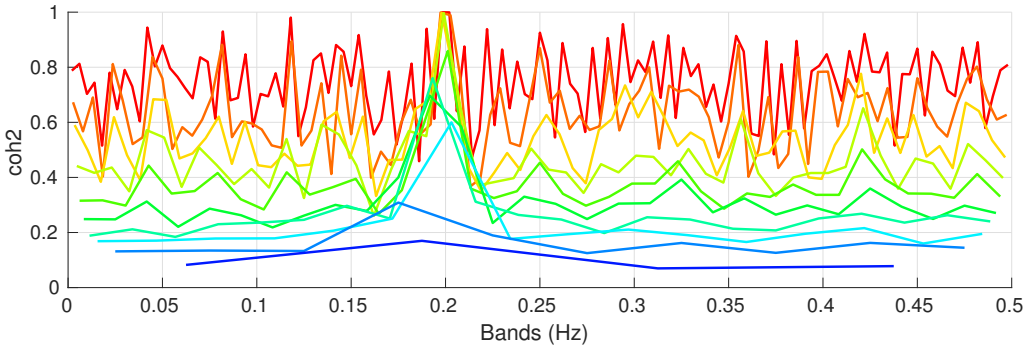


Figure 3.3: Coherence dependence with band width dfB (warmer colour: narrower bands, colder colour: wider bands).

of coherence changes).

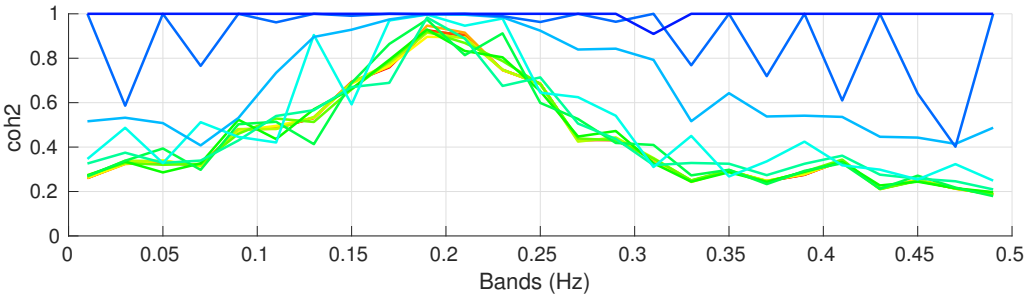


Figure 3.4: Coherence dependence with DFT resolution df (warmer colour: more resolution, colder colour: less resolution).

The tests of dfB and nra are based on characterize the suitability of the coherence obtained. To do so a combination of three different metrics is used. One is the maximum value of the coherence (the higher the coherence, the best the results of the inversion). The second is the difference between the maximum and a reference value, intended to characterize the noise level, and it is computed as a certain quantile of data (something between the median and the 95% percentile). The third one is the ratio between a magnitude characterizing the neighbourhood of the maximum (the minimum of its nearest neighbours, to characterize if the maximum corresponds only a spark or if it really has a peak shape) and another magnitude representing the rest of points (coherence average). These three metrics must be combined so that if one one of them is bad, then that parameter value will not be chosen. The way to combine them is to normalize them between zero and one and multiply them together to get the combined metric. Since they three change rapidly, they are first low-pass filtered. This procedure results in coherence functions that seem reliable in order to identify wave presence. The resulting well-shaped coherence is also used to preselect bands, for the case cBathy is told to use them.

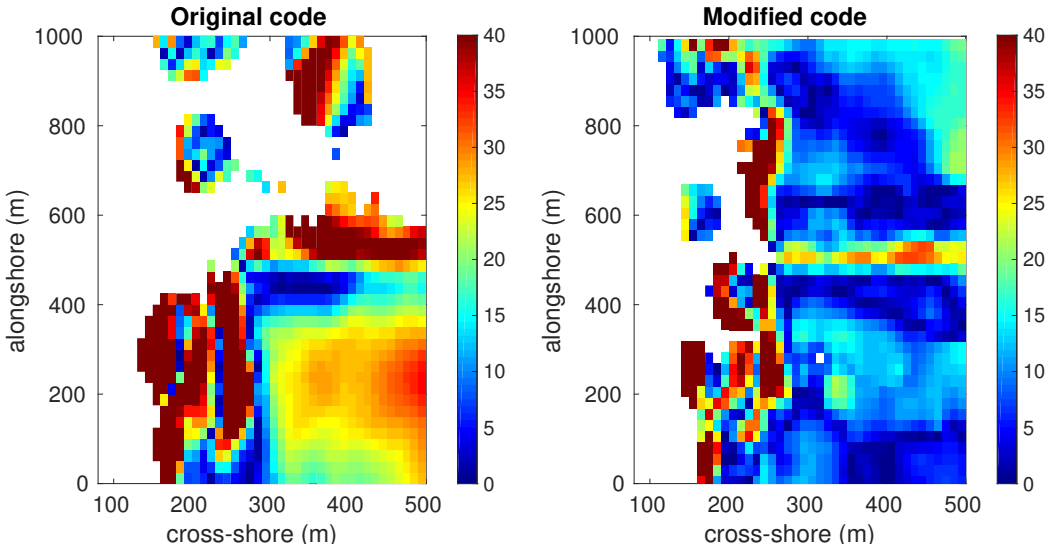


Figure 3.5: Comparison of the bathymetric inversion errors for the cBathy demo case. coast is to the left of the images, the colorbar is in relative error [%].

Since the effect of $df\,B$ and nra overt the coherence are coupled, their tests are iterated alternatively. This is, the test starts with some predefined values (with a high value for df so it does not affect the other tests), and changes $df\,B$ to find the value that maximizes the combined metric. Then $df\,B$ is fixed, and nra is tuned to generate also a maximum of the combined metric. This process is repeated some times (between 5 and 10) to allow both parameters to converge to their proper values. Finally, once these two parameters are selected, df is changed until the coherence starts to change more than a given threshold. In this way, proper values for the three frequential parameters are obtained.

The demo case that cBathy authors provide with the code has been inverted using the original routines and also using the routines that incorporate the internal modifications implemented on cBathy and also the external parameter tuning. Results, shown in Figure 3.5, clearly prove that the modified code improves the performance significantly.

3.1.4 Proposals for further changes on cBathy

The most important modification that could be implemented on cBathy if more time was available are the following:

1. **Continuous coherence**. The essential problem about using bands is that although they reduce the noise present in the spectra, they also worsen the resolution in the frequency domain, so that the frequencies estimated may differ from the real ones. However, other methods can be applied to reduce the noise present in the coherence.

For example:

2. **Correlations with amplitude**. cBathy uses phase correlation to find coherent bands and to extract the phase vector. However, the usual spectral correlation involving amplitudes might be used instead. In order to average it through the band, all spectrum amplitudes should be considered to produce a coherence that is between zero and one. The usual way would be to normalize the spectra individually (as when one computes the angle between two vectors).

$$G_m(f) = \mathcal{F}\left\{s_m(t)\right\}(f) = \left|G_m(f)\right| \cdot e^{j\angle G_m(f)}$$

$$CSM_{m,n,b} = \frac{\sum_{f_l \in band_b} \overline{G_m(f_l)} G_n(f_l)}{\sqrt{\sum_{f_l \in band_b} \left|G_m(f_b)\right|^2} \sqrt{\sum_{f_l \in band_b} \left|G_n(f_b)\right|^2}}$$

A couple of tests have been run using it. The resulting coherence is a bit more noisy but also presents peaks that protrude more over the noise floor. However, the inversion itself fails. More tests need to be run, and their results studied.

   (a) Welch's method. The temporal signal would be split in smaller pieces and the DFT of each fragment would be computed separately. Then an average using the different realizations would be performed. This could be done for phase correlation

$$CSM^{phase}_{m,n,b} = \frac{1}{L} \sum_{l=1}^{L} \overline{G^l_m(f)} G^l_n(f)$$

   or for the spectral correlation involving amplitudes.

$$CSM_{m,n,b} = \frac{\sum_{l=1}^{L} \overline{G^l_m(f)} G^l_n(f)}{\sqrt{\sum_{l=1}^{L} \left|G^l_m(f)\right|^2} \sqrt{\sum_{f=1}^{L} \left|G^l_n(f)\right|^2}}$$

   where $CSM^{phase}_{m,n,b}$ represents a cross-spectral matrix using phase correlation, $CSM_{m,n,b}$ symbolizes a cross-spectral matrix using correlations that include amplitude, L is the number of fragments in which the temporal signals are split, and l is the index that lists fragments. Notice that using fragments that span smaller time intervals would result in a rectangular windowing that would provoke wider sincs. Therefore, wave peaks could mask smaller signals that lie near. However, the frequencies could be determined with more precision because the resolution of the spectrum would be that of the pixel DFT.

(b) Coherence filtering. Instead of filtering the CSM by averaging along bands, it could be smoothed by means of a sliding window (convolution). Due to the width of the filter response, the effects of one frequency might affect other ones, but since the resolution would be that of the pixel DFT, the precision would be improved.

These new ways of computing coherence would present a new problem. Selecting the most coherent frequencies would not be as easy as picking those with the biggest value of the coherence, since in this way only frequencies around biggest peak in the coherence would be selected. Therefore, a new algorithm peak finding algorithm would be needed.

3. **Wave-vector fitting improvement**. The global phase of the phase array is set so that the data point closest to the centre of the tile has a phase term of 1 (see Subsection 3.1.2). Thereby the expression of the wave spatial phase term can be rewritten as;

$$\exp\left(-j\left(\vec{k}\cdot\vec{r}+\phi_0\right)\right) \rightarrow \exp\left(j\left(-\vec{k}\cdot(\vec{r}-\vec{r}_{cc})\right)\right) \tag{3.7}$$

Where $\vec{r}_{cc}$ are the coordinates of closest point to the tile center (whose phase is set to one). In this way, the fitting could be performed using two parameters, *i. e.*, without ϕ_0. Another interesting change that could be done is to consider the possibility of wave reflection. Then, instead of fitting one exponential, two of them should be used, with different amplitudes, and different angles of the wave-vector (but the same wavenumber).

$$\theta_i[m] = -k\cos(\alpha_i)(x_m - x_{cc}) - k\sin(\alpha_i)(y_m - y_{cc}) + \phi_{cc,i} \tag{3.8}$$
$$\theta_r[m] = -k\cos(\alpha_r)(x_m - x_{cc}) - k\sin(\alpha_r)(y_m - y_{cc}) + \phi_{cc,r} \tag{3.9}$$

This means an optimization with six parameters, with a constraint arising from the condition that the global phase is a free parameter.

$$v[m] = \frac{1}{1+\rho^2+2\rho\cos\left(\theta_r-\theta_i\right)}\left(e^{j\theta_i[m]}+\rho e^{j\theta_r[m]}\right) \tag{3.10}$$
$$\frac{1}{1+\rho^2+2\rho\cos\left(\phi_{cc,r}-\phi_{cc,i}\right)}\left(e^{j\phi_{cc,i}}+\rho e^{j\phi_{cc,r}}\right) = 1 \tag{3.11}$$

Where m represents the index of the pixel inside the tile, θ_i and θ_r are the angles of the modelled incident and reflected waves' spatial phases, x_{cc} and y_{cc} are the

coordinates of the point whose phase is set to one (constraint), and k, α_i, $\phi_{cc,i}$, ρ, α_r and $\phi_{cc,r}$ the parameters to optimize in order to fit the modelled phase.

3.1.5 Kalman filtering

All the explanation presented regarding cBathy has only regarded the single inversion process, this is, obtaining a bathymetry using just one video. However, cBathy consists in a three steps process, where the last part is fusing data from different single inversions to get a better estimate. It achieves that by using a Kalman filter over successive depth estimates. In fact, the authors of cBathy point out that this is the most important point to the performance and success of cBathy. The single video inversion process has been improved, but it is mandatory to wrap this single video inversion process with Kalman filtering routine in the future. However, in order to implement a Kalman filter, a model to describe the degradation of estimates is needed (it is beach dependent), and there are few directives in the literature on how to build such an estimate.

3.2 uBathy

UBathy is a bathymetric inversion method which is being developed by the directors of this project. It also uses Airy dispersion equation to find local depth using frequencies and wavenumbers of the monochromatic components present in a video of the coastal waves. In order to decouple the wave signal into its components, uBathy uses empirical orthogonal functions (EOFs).

The signal under study is modelled as shown by Equation 3.1. The form of the spatial phase $\phi(\vec{r})$ is

$$\phi_c(\vec{r}) = -\phi_0 - \int_{\vec{0}}^{\vec{r}} \vec{k}_c \cdot \vec{dr} \tag{3.12}$$

The objective of the decoupling process is to obtain the each component phase separately, in order to have information to estimate all the frequencies and wavenumbers. To do so, uBathy relies on the orthogonality between sinusoids of different repetition periods, using EOFs. EOF decomposition allows to obtain a set of orthogonal temporal functions $w_c(t)$ and a set of orthogonal spatial functions $v_c(\vec{r})$ which can be used to rewrite the spatiotemporal decoupled signal $q(\vec{r}, t)$ as shown in Equation 3.13.

$$q(\vec{r},t) = \sum_c u_c(t) v_c(\vec{r}) \tag{3.13}$$

The signal under study can present the same functional form if the cosine of each monochromatic component is split. EOF uses.

$$s_c(\vec{r},t) = A_c(\vec{r}) \cos\left(\phi_c(\vec{r})\right) \cos\left(2\pi f_c t\right) - A_c(\vec{r}) \sin\left(\phi_c(\vec{r})\right) \sin\left(2\pi f_c t\right)$$

However, this equation presents two inconveniences. The first one is that each monochromatic wave present in the inversion domain is retrieved twice, and the computations for each of them are repeated. The second inconvenience is that the phase variations from the spatial term can not be retrieved, since in the data is not possible to distinguish between the amplitude and the cosine. To avoid this situation, the signal is represented in the complex plane, using phasors. To this effect, the temporal Hilbert transform of the signal ($\hat{s}_c(\vec{r},t)$ is computed, and used to build the corresponding analytic signal (a_{s_c}).

$$\hat{s}_c(\vec{r},t) = A_c(\vec{r}) \cos\left(\phi_c(\vec{r})\right) \sin\left(2\pi f_c t\right) + \sin\left(\phi_c(\vec{r})\right) \cos\left(2\pi f_c t\right) \tag{3.14}$$

$$a_{s_c}(\vec{r},t) \equiv s_c(\vec{r},t) + j\ \hat{s}_c(\vec{r},t) = A_c(\vec{r})\ \exp\left(j2\pi f_c t\right)\ \exp\left(j\phi_c(\vec{r})\right) \tag{3.15}$$

The analytic signal still preserves the form needed by EOF decomposition, but presents only one spatiotemporal function pair for each monochromatic tone. It also allows to obtain spatial phases, by computing the angle of the spatial function obtained by EOF decomposition.

3.2.1 Orthogonality check

However, in order to ensure that the results of the decomposition are the wave terms expected by Equation 3.15, the different temporal functions and spatial functions need to be orthogonal. The lack of orthogonality can arise by the fact that spatial terms are not monochromatic, but changing wavelength locally. It can also be caused by the windowing of the signal (finite observation time and spatial scope) or by its discretization (sampling). It is even possible that the signal contains two monochromatic components with the same frequency, but for the moment this case will not be considered.

The orthogonality of the temporal terms is checked in Derivation 3.2) and the orthogonality of the spatial terms in Derivation 3.3. The checking consists in computing the scalar product between functions of different monochromatic components. The inner product is denoted by $\langle a(x_q), b(x_q)\rangle = \sum_q \overline{a(x_q)} b(x_q)$.

Temporal terms orthogonality

The temporal signals are sampled, *i. e.*, its values are only known for times $t_n = \mathrm{n} \cdot T_s = \mathrm{n} \ / \ f_s$, ($\mathrm{n} \in \mathbb{N}$, between 0 and N).

$$w_c[n] = \frac{1}{\sqrt{N}} \exp\left(j\, 2\pi f_c t_n\right) \tag{3.16}$$

where $w_c[n]$ stands for the discrete temporal signal associated with the monochromatic component of index c, properly normalized.

$$\langle w_{c_1}\, ,\, w_{c_2}\rangle = \frac{1}{N} \sum_{n=0}^{N-1} \exp\left(j\, 2\pi \left(f_{c_2} - f_{c_1}\right) t_n\right)$$

$$\frac{1}{N} \sum_{n=0}^{N-1} \exp\left(j\, 2\pi \left(f_{c_2} - f_{c_1}\right) t_n\right) = \frac{1}{N}\, \frac{\exp\left(j\, 2\pi N\ \left(f_{c_2} - f_{c_1}\right)/f_s\right) - 1}{\exp\left(j\, 2\pi \left(f_{c_2} - f_{c_1}\right)/f_s\right) - 1}$$

$$\frac{1}{N}\, \frac{\exp\left(j\, 2\pi N \left(f_{c_2} - f_{c_1}\right)/f_s\right) - 1}{\exp\left(j\, 2\pi \left(f_{c_2} - f_{c_1}\right)/f_s\right) - 1} = \frac{1}{N}\, \frac{\sin\left(N\pi \left(f_{c_2} - f_{c_1}\right)/f_s\right)}{\sin\left(\pi \left(f_{c_2} - f_{c_1}\right)/f_s\right)}\ \ e^{j\,\pi(N-1)\left(f_{c_2}-f_{c_1}\right)/f_s}$$

$$\left|\frac{1}{N}\, \frac{\sin\left(N\pi \left(f_{c_2} - f_{c_1}\right)/f_s\right)}{\sin\left(\pi \left(f_{c_2} - f_{c_1}\right)/f_s\right)}\right| \leq \begin{cases} 1 & , f_{c_2} - f_{c_1} < \frac{f_s}{N\pi} \\ \frac{f_s}{N\pi\left(f_{c_2}-f_{c_1}\right)} + \frac{f_s}{N\pi\left(1-\left(f_{c_2}-f_{c_1}\right)\right)} & , f_{c_2} - f_{c_1} > \frac{f_s}{N\pi} \end{cases}$$

$$\frac{f_s}{N\pi \left(f_{c_2} - f_{c_1}\right)} + \frac{f_s}{N\pi \left(1 - \left(f_{c_2} - f_{c_1}\right)\right)} \sim \frac{f_s}{N\pi \left(f_{c_2} - f_{c_1}\right)}$$

This piecewise function can be rewritten considering that in its interval, the current expression is smaller than the other one. Then:

$$\left|\langle w_{c_1}\,,\, w_{c_2}\rangle\right| = \min\left(1\ ,\ \frac{f_s}{N\pi\,(f_{c_2}-f_{c_1})}\right) \tag{3.17}$$

The ratio of the inner product between temporal functions of different monochromatic component and the square of their norm can be used as a measure of orthogonality. The w_c arrays are normalized, and therefore this ratio is exactly the value shown in Equation 3.17.

This ratio is directly proportional to the sampling frequency, and inversely proportional to the number of temporal samples of our signals. Therefore, the orthogonality between temporal phase terms can be improved by sampling faster, or by recording longer videos.

Derivation 3.2: Monochromatic components temporal phase orthogonality check.

Spatial terms orthogonality

The input signals are measured in a discrete set of spatial positions, corresponding to each of the pixels.

$$v_c^{'}[m] = A_c(\vec{r}_m)\ \exp\left(j\,\phi_c(\vec{r}_m)\right)$$

Here, $v_c^{'}[m]$ are the discrete spatial signal associated with the monochromatic component of index c, and m is the index that lists the pixels. First of all, they need to be normalized. Since the phase term has amplitude one, the norm will only depend on the amplitude. Therefore, the RMS amplitude can be used to normalize the spatial array.

$$A_c^{\text{rms}} = \sqrt{\frac{1}{M}\sum_m \left(A_c(\vec{r}_m)\right)^2} = \frac{1}{\sqrt{M}}\sqrt{\left(A_c(\vec{r}_0)\right)^2 + \left(A_c(\vec{r}_1)\right)^2 + \ldots + \left(A_c(\vec{r}_{M-1})\right)^2} \tag{3.18}$$

$$\|v_c'[m]\| = \sqrt{\sum_m \left(A_c(\vec{r}_m)\right)^2} = \sqrt{M} A_c^{\text{rms}}$$

Thus, the properly normalized spatial signal $v_c[m]$ can be written as:

$$v_c[m] = \frac{1}{\sqrt{M} A_c^{\text{rms}}} A_c(\vec{r}_m)\ \exp\left(j\,\phi_c(\vec{r}_m)\right) \tag{3.19}$$

And then the inner product between two of this signals can be computed.

$$\langle v_{c_1}\,, v_{c_2}\rangle = \frac{1}{M A_{c_1}^{\text{rms}} A_{c_2}^{\text{rms}}} \sum_m A_{c_1}(\vec{r}_m) A_{c_2}(\vec{r}_m)\ \ \exp\left(-j\left(\phi_{c_2}(\vec{r}_m) - \phi_{c_1}(\vec{r}_m)\right)\right)$$

Since dependence of amplitudes and spatial phases on position is unknown, the exact value of this inner product can not be obtained. However, its magnitude can still be estimated somehow. In order to do this, the square of its absolute value is computed (the square is easier to manipulate, cause it is directly a sum of terms, without a square root).

Define $\theta_{c1,c2}(\vec{r}_m) = \phi_{c_2}(\vec{r}_m) - \phi_{c_1}(\vec{r}_m)$, to ease writing.

$$\left|\langle v_{c_1}\,, v_{c_2}\rangle\right|^2 = \frac{1}{\left(M A_{c_1}^{\text{rms}} A_{c_2}^{\text{rms}}\right)^2} \sum_{m_1} \sum_{m_2} A_{c_1}(\vec{r}_{m_1}) A_{c_1}(\vec{r}_{m_2}) A_{c_2}(\vec{r}_{m_1}) A_{c_2}(\vec{r}_{m_2})$$
$$e^{-j\theta_{c1,c2}(\vec{r}_{m_2})}\ e^{j\theta_{c1,c2}(\vec{r}_{m_1})}$$

Then, if all the amplitudes that a monochromatic wave presents in the inversion domain are of the same order of magnitude (the may vary about a factor two), the inner product can be computed as an incoherent sum of complex exponentials. To support this assumption, one can refer to the theoretical amplitudes predicted for FUNWAVE by the method described in Simarro [2013] (Figure 3.6), or also to the amplitudes observed in videos of purely monochromatic cases (Figure 3.7).

Therefore, it is assumed that the position dependent amplitudes can be replaced by their rms equivalent without altering the order of magnitude of the inner product.

$$\left|\langle v_{c_1}\,,v_{c_2}\rangle\right|^2 \sim \frac{1}{\left(MA_{c_1}^{\text{rms}}A_{c_2}^{\text{rms}}\right)^2}\sum_{m_1}\sum_{m_2}\left(A_{c_1}^{\text{rms}}A_{c_2}^{\text{rms}}\right)^2\;\; e^{-j\theta_{c1,c2}(\vec{r}_{m_2})}\;\; e^{j\theta_{c1,c2}(\vec{r}_{m_1})} \;=$$

$$= \;\frac{1}{M^2}\sum_{m_1}\sum_{m_2} e^{-j\left(\theta_{c1,c2}(\vec{r}_{m_2})-\theta_{c1,c2}(\vec{r}_{m_1})\right)}$$

The difference "$\theta_{c1,c2}(\vec{r}_{m_2}) - \theta_{c1,c2}(\vec{r}_{m_1})$" becomes exactly one when $m_1 = m_2$, and its angle presents all the possible values with the same empirical frequency (it is uniformly distributed). Then, the double sum can be split in a coherent sum of ones (case $m_1 = m_2$, with result M) and an incoherent sum of $(M^2 - M)$ complex exponentials (case $m_1 \neq m_2$, with a result of $\sqrt{M^2 - M}$).

$$\frac{1}{M^2}\sum_{m_1}\sum_{m_2} e^{-j\left(\theta_{c1,c2}(\vec{r}_{m_2})-\theta_{c1,c2}(\vec{r}_{m_1})\right)} \simeq \frac{1}{M^2}\left(M+\sqrt{M^2-M}\right) \simeq$$

$$\simeq \frac{1}{M^2}\left(M+M\right) = \frac{2}{M}$$

$$\left|\langle v_{c_1}\,,v_{c_2}\rangle\right|^2 \sim \frac{2}{M} \sim \frac{1}{M}$$

Since the $v_c[m]$ arrays are normalized, it is easy to see that the ratio between different waves inner product and same waves inner product is:

$$\frac{\left|\langle v_{c_1} v_{c_2}\rangle\right|}{\left|\langle v_{c_1} v_{c_1}\rangle\right|} \sim \frac{1}{\sqrt{M}} \tag{3.20}$$

Again, this ratio can be used as a measure of orthogonality. It becomes smaller with bigger meshes, but not as fast as the temporal phase term case.

The inner product between the analytic signal spatial terms of the different purely monochromatic wave cases studied has been computed, having amplitudes whose order of magnitude corresponds to the estimation presented ($1/\sqrt{M}$).

Derivation 3.3: Monochromatic components spatial phase orthogonality check.

Therefore, by using enough points in time and space, and using a resolutive enough grid (to improve the temporal terms orthogonality, but also to avoid aliasing), the temporal terms of different monochromatic waves are orthogonal, as well as the spatial terms. Then,

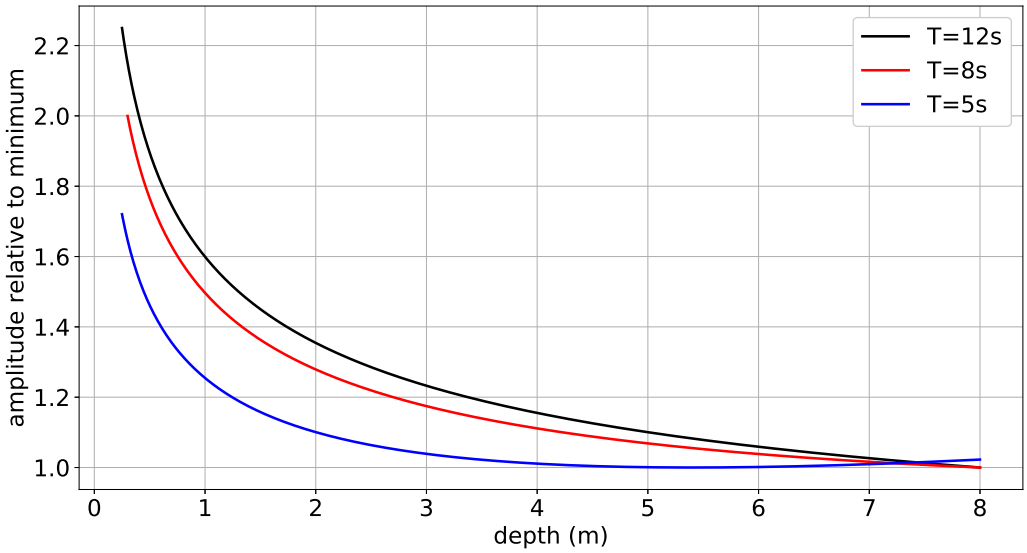


Figure 3.6: Predicted amplitude vs. depth relationship for FUNWAVE equations.

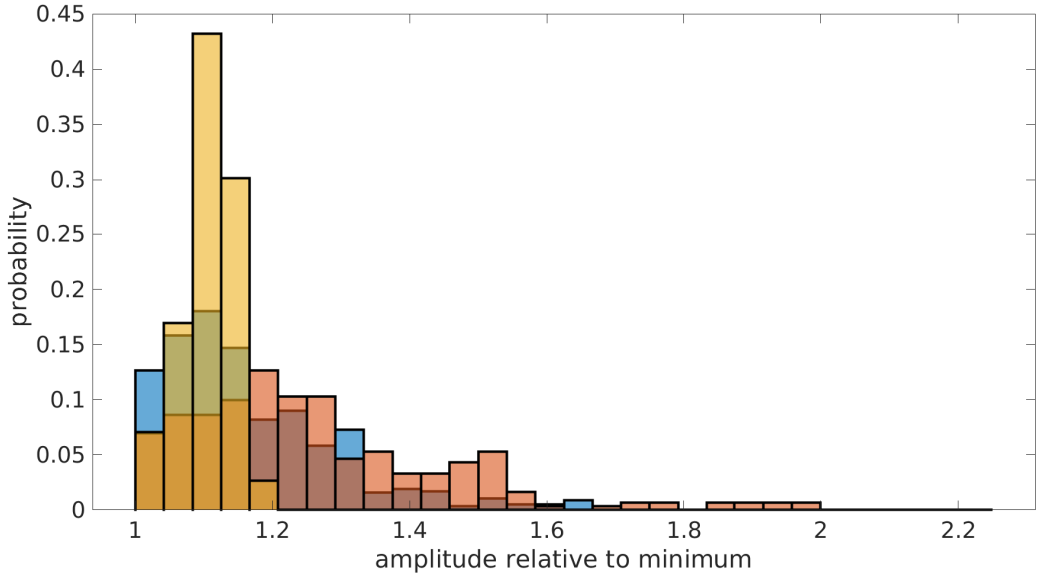


Figure 3.7: Histogram of amplitudes for one of the decoupled monochromatic waves.

the results of EOF decomposition can be assimilated to those monochromatic temporal and spatial terms (EOF decomposition results are orthogonal up to the machine epsilon, but if data was able to present orthogonal enough components, those and the EOF decomposition results are expect to differ only in a small amount).

3.2.2 EOF decomposition process

The EOF decomposition works by computing the covariance matrix between pixel intensities. This is, each pixel is understood as a random variable, and each image corresponds to a realization. The covariance matrix can also be interpreted as the Gramian that form the demeaned temporal series of each pixel. Then, the first step is to extract the temporal means from the measured signal $x(\vec{r}_m, t_n)$.

$$s(\vec{r}_m, t_n) = x(\vec{r}_m, t_n) - \sum_{n=0}^{N-1} x(\vec{r}_m, t_n)$$

Assuming that the demeaned signal has the modelled structure shown in Equation 3.1, and applying Equations 3.16, 3.17, 3.19, 3.20, the covariance matrix can be computed as:

$$\begin{aligned} s_c(\vec{r}_m, t_n) &= \sum_c A_c(\vec{r}_m)\ \exp\left(j\,\phi_c(\vec{r}_m)\right)\ \exp\left(j\,2\pi f_c t_n\right) = \\ &= \sqrt{MN}\ A_c^{rms}\ v_c[m]\ w_c[n] \end{aligned}$$

$$\begin{aligned} C_{m_1,m_2} &= \sum_{n=0}^{N-1} \overline{a_s(\vec{r}_{m_1}, t_n)}\ a_s(\vec{r}_{m_2}, t_n) = \\ &= \sum_{n=0}^{N-1} \left(\sum_{c_1} \overline{s_{c_1}(\vec{r}_{m_1}, t_n)} \right) \left(\sum_{c_2} s_{c_2}(\vec{r}_{m_2}, t_n) \right) = \\ &= MN \sum_{n=0}^{N-1} \left(\sum_{c_1} A_{c_1}^{rms}\ \overline{v_{c_1}[m_1] w_{c_1}[n]} \right) \left(\sum_{c_2} A_{c_2}^{rms}\ v_{c_2}[m_2] w_{c_2}[n] \right) = \\ &= MN \sum_{c_1,c_2} \left(A_{c_1}^{rms} A_{c_2}^{rms}\ \overline{v_{c_1}[m_1]} v_{c_2}[m_2] \left(\sum_{n=0}^{N-1} \overline{w_{c_1}[n]} w_{c_2}[n] \right) \right) = \\ &= \sum_{c_1,c_2} MN\ A_{c_1}^{rms} A_{c_2}^{rms}\ \overline{v_{c_1}[m_1]} v_{c_2}[m_2]\ \langle w_{c_1}\,,\, w_{c_2} \rangle \xrightarrow{\langle w_{c_1}\,,\, w_{c_2}\rangle = \delta_{c_1,c_2} (\text{Der. } 3.2)} \\ &\xrightarrow{\langle w_{c_1}\,,\, w_{c_2}\rangle = \delta_{c_1,c_2} (\text{Der. } 3.2)} \sum_c \left(MN\,(A_c^{rms})^2 \right)\ \overline{v_c[m_1]} v_c[m_2] \end{aligned} \tag{3.21}$$

Then, the eigenvectors of this matrix are extracted. Following Derivation 3.3, it can be assumed that the $v_c[m]$ are orthonormal. Therefore, the last line of Equation 3.21 shows directly the eigendecomposition of the covariance matrix (Equations 3.22 and 3.23 rewrite it to show it more visually).

$$\lambda_c = MN\,(A_c^{rms})^2 \tag{3.22}$$

$$\underline{\underline{C}} = \sum_c \lambda_c\ \underline{v_c}\ \underline{v_c}^H \tag{3.23}$$

The eigenvectors (called empirical orthogonal functions or modes) can be interpreted as a set of linear transformations over the pixels' temporal series, in such a way that the resulting temporal series (principal components) have a covariance matrix which is diagonal (it is equivalent to say that each pixel temporal series is a linear combination of the principal components defined by the eigenvectors). The variances of that matrix correspond to the eigenvalues of the extracted eigenvectors. They can be interpreted as the variance that each of the eigenvectors contributes with to the total variance. The principal components can be obtained by applying these linear combinations to the set of original temporal series. This is equivalent to compute the inner product of the total spatiotemporal signal over each mode.

$$\begin{aligned}
w_c[n] &= \frac{1}{\lambda_c}\langle v_c\,,\ s(\vec{r}_m,t)\rangle = \sum_m \overline{v_c[m]}\ s(\vec{r}_m,t) = \\
&= \frac{1}{\lambda_c}\sum_m\sum_\gamma \overline{v_c[m]}\ \sqrt{MN}\ A_\gamma^{rms}\ v_\gamma[m]\ w_\gamma[n] = \\
&= \frac{1}{\lambda_c}\sum_\gamma \sqrt{MN}\ A_\gamma^{rms}\ w_\gamma[n]\ \left(\sum_m \overline{v_c[m]} v_\gamma[m]\right) \xrightarrow{\langle v_{c_1}\,,\,v_{c_2}\rangle=\delta_{c_1,c_2}(\text{Der. 3.3})} \\
&\xrightarrow{\langle v_{c_1}\,,\,v_{c_2}\rangle=\delta_{c_1,c_2}(\text{Der. 3.3})} \frac{1}{\lambda_c}\ \sqrt{MN}\ A_c^0\ w_c[n] = w_c[n]
\end{aligned} \tag{3.24}$$

Once the EOF decomposition process has been presented, it is mandatory to state that the covariance matrix and the inner products described by Equation 3.24 do not need to be computed. There is an specific matrix decomposition, the singular value decomposition (SVD), that directly decouples the demeaned data matrix into three matrices $\mathbf{W}$, $\mathbf{\Lambda}$ and $\mathbf{V}$ that contain, respectively, the temporal phase terms by columns, the eigenvalues in the diagonal, and the spatial terms by columns. This factorization is of the form:

$$Y_{m,n} = s(\vec{r}_m, n)$$

$$\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{W}^H$$

There are algorithms that compute it quite fast, saving time and memory and obtaining more accuracy, compared to the covariance matrix direct computation. As a final remark about decoupling, it is interesting to note that the EOF decomposition also provides some rejection to noise and interference, cause it may extract them as another pair of spatial and temporal vectors. The results conducted over real coastal data seem to support this hypothesis.

3.2.3 Wavenumber extraction

Once the measured signal has been decoupled, the monochromatic temporal and spatial phases are used to extract frequencies and wavenumbers. The temporal phase terms are just constant frequency 1D oscillations. The literature is full with estimators for that type of functions, and thus the frequency extraction does not present any problem. On the contrary, the spatial phase term is a 2D oscillation with a varying wave-vector. A local estimation for the wavenumber is needed, and this requires a subtler approach. From now on $v_c[m]$ redefined to contain only phase information (by dividing each sample by its amplitude).

The original approach to wavenumber estimation consists in a linear fitting over the phase angle. For each sample of the spatial phase, a neighbourhood of a certain user-defined radius is selected. Then, all the samples inside the neighbourhood are referred to the phase of the central sample, to avoid angle jumps inside the neighbourhood (see Figure 3.8).

$$\hat{v}[q] = \text{angle}\left(v[q]\ \overline{v[m]}\right) \quad , \quad \text{q} \in \text{neighbourhood of } \vec{r}_m$$

Using the angles of the normalized phases inside the neighbourhood, a polynomial can be fitted, in order to mimic the Taylor expansion of the spatial phase around the neighbourhood centre. The wave-number is formed by taking the coefficients of the linear terms.

$$v_c[m] \;=\; \theta_c - \int_{\vec{0}}^{\vec{r}} \vec{k_c} \cdot \vec{dr} \;\simeq\; \theta_c - k_x[m]x_m - k_y[m]y_m + o\left(x_m^2, x_m y_m, y_m^2\right)$$

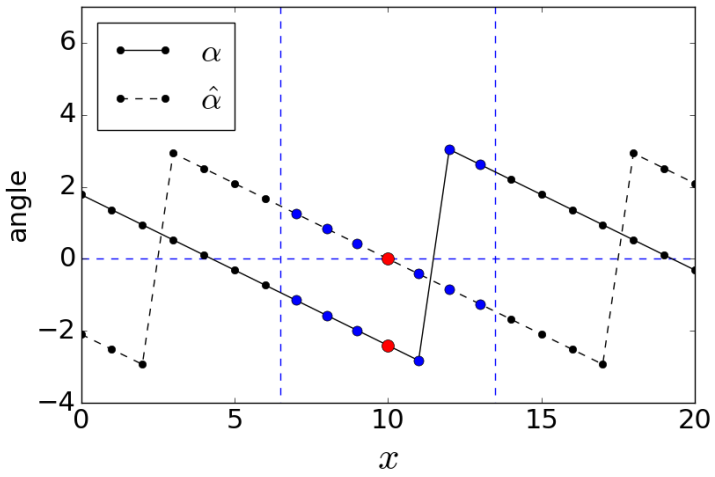


Figure 3.8: Result of the phase normalization.

where coordinates are referred to the neighbourhood centre. The wavenumber obtained is used to find a characteristic wavelength, which in turn is used to redefine the size of the neighbourhood, repeating the process to get a better estimate (the first neighbourhood may be too big and contain angle jumps).

However, this method of wavenumber extraction fails if consider reflection is considered. Until this point, is has been assumed that all the monochromatic waves which build up the signal under study have different frequencies. But if a reflected wave appears, this is no longer true. In this case, the EOF decomposition cannot distinguish between the incident and the reflected waves, since they are not orthogonal. Then, the returned spatial phase will be a superposition of them. Therefore, its angle will no longer provide wave phase information directly, preventing the use of linear fittings. However, a nonlinear fitting can be conducted over the sum of the two complex exponentials, as follows:

$$\theta_{c,i}[m] = -k_c \cos(\alpha_{c,i}) x_m - k_c \sin(\alpha_{c,i}) y_m + \theta_{c,i} \tag{3.25}$$

$$\theta_{c,r}[m] = -k_c \cos(\alpha_{c,r}) x_m - k_c \sin(\alpha_{c,r}) y_m + \theta_{c,r} \tag{3.26}$$

$$v[m] = \frac{1}{1 + \rho^2 + 2\rho \cos\left(\theta_r - \theta_i\right)} \left(e^{j\theta_i[m]} + \rho e^{j\theta_r[m]} \right) \tag{3.27}$$

$$\frac{1}{1 + \rho^2 + 2\rho \cos\left(\phi_{cc,r} - \phi_{cc,i}\right)} \left(e^{j\phi_{cc,i}} + \rho e^{j\phi_{cc,r}} \right) = 1 \tag{3.28}$$

This alternative method has been implemented successfully, allowing to retrieve the wavenumber even in presence of strong reflections.

4 . Results

4.1 Methodology

The videos of waves corresponding to cases described in section 2.3 have been inverted using cBathy and uBathy. Since the exact bathymetries are known, the absolute value of the difference between the inverted and the exact bathymetry can be defined (hereinafter absolute error), and also its value relative to the local exact depth (hereinafter relative error).

$$\epsilon_{abs} = |h_{inv} - h_{ex}|$$

$$\epsilon_{rel} = \frac{\epsilon_{abs}}{h_{ex}}$$

Here, ϵ_{abs} is the absolute error, ϵ_{rel} is the relative error, h_{inv} is the inverted depth and h_{ex} is the exact depth. The inversion relative errors can be computed and displayed using a colour code. An example of the inversion results is shown in Figure 4.1.

All the inversion relative errors have been computed and depicted, saturated to 10% of pixel depth error in order to present a common scale that allows to compare different images and presenting an easier characterization of the results. They are presented in the Section 4.2.

Also, for each of the inversions performed, the obtained errors can be sorted by magnitude, defining percentiles (the n-th error percentile is the value that separates the $n\%$ smallest errors and the remaining $(100 - n)\%$), and quartiles (the first quartile, Q1, corresponds to the percentile 25, the second quartile, Q2, also known as the median, corresponds to the percentile 50, and the third quartile, Q3, corresponds to the percentile 75). The percentiles can be used to characterize how the errors of each inversion are distributed, and also can be used to clip the data in order to neglect outliers. The root mean square error clipped to the percentile 95 has been defined as

$$RMS_{95} = \sqrt{\frac{1}{M}\sum_{m} \epsilon_{abs}}$$

where RMS_{95} is the root mean square error clipped to the percentile 95, m is an index listing the points of pixels of each image (points where the bathymetry has been inverted), and M is the number of pixels. For each inversion performed, the quartiles of the absolute

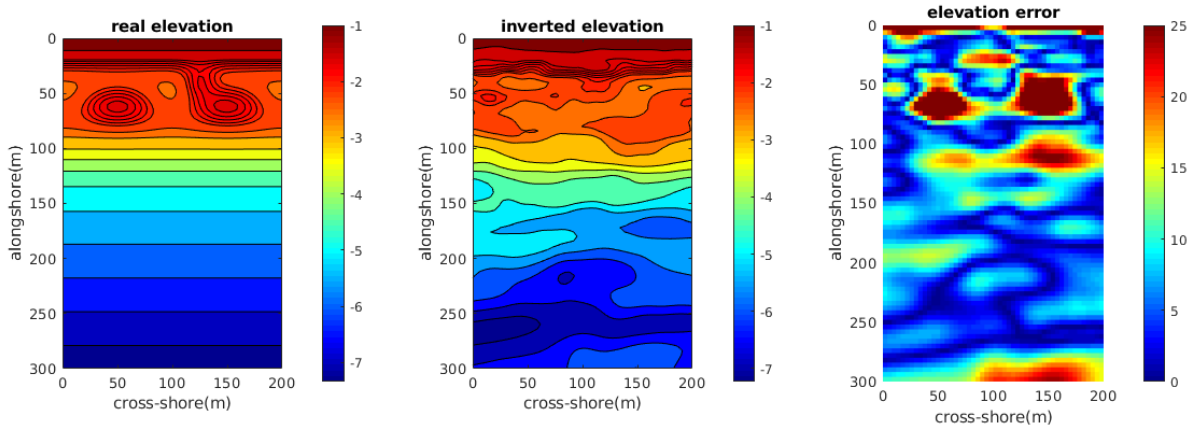


Figure 4.1: Results of one of the inversions. The bathymetry used is the one with structures, the wave case is the polychromatic one, with amplitude factor of x0.1. The video used for the inversion has a sampling frequency of 2.0 Hz and a spatial resolution of 0.25 ppm. The inversion was performed using cBathy. The figure shows, from left to right, the original bed elevation in metres, the inverted bed elevation in metres and the resulting relative error in % (bed elevation is defined as the opposite of depth). The horizontal axis is alongshore in metres, and the vertical one represents cross-shore in metres. The coast is up.

error and its RMS_{95} error have been computed. These quantities are presented in Section 4.3.

4.2 Results

The results of the inversions have been ordered according to the evolution of the parameters studied.

The evolution with spatial resolution is quite clear for cBathy. The results are similar for bigger resolutions, (0.50 ppm and 0.25 ppm), although they seem a bit better for the case of 0.25 ppm. Using a low resolution video (0.1 ppm) causes greater errors in the inversion, specially onshore (Table 4.1). Regarding the sampling frequency used in the videos, cBathy presents two different behaviours (Table 4.1). For the monochromatic cases, and near the coast for the polychromatic cases, results worsen with lower sampling frequency, specially under 1 Hz. However, offshore errors seem to diminish with smaller sampling frequency. Therefore, the optimum resolutions for cBathy are near 0.25 ppm and 2.0 Hz, although this inversion method can work with a reduced performance with other resolutions. These values reassure the resolutions suggested by Holman and Haller [2013].

The same evolution with spatial resolution is observed in the uBathy results (Table 4.2). However, when using high resolution and low sampling frequency video, uBathy fitting routines are unable to find wavenumber for most or any of the points, resulting in a generalized failure of the inversion. UBathy results are quite robust to changes in the sampling frequency, with the exception the cases where it fails (Table 4.2). Then, the optimum resolutions for uBathy are also 0.25 ppm and 2.0 Hz. However, uBathy can work with almost the same performance with other resolutions.

The dependence with the bathymetry type is also clear. Errors are quite noticeable over the bumps and undulations of the cases with structures (Table 4.3). UBathy presents some small error undulations offshore, probably caused by reflections and the associated fitting. CBathy presents two extra zones of error next to the bumps, the cause for which is unknown.

Both cBathy and uBathy get worse results with bigger amplitudes (Table 4.3). There is a big change between the linear modelled case and the FUNWAVE modelled one. Then, between FUNWAVE modelled cases, the effect seems to worsen faster for bigger amplitudes (this is, the change is subtler between x0.1 and x0.4 than between x0.4 and x1.0). The more affected areas are those closer to the coast. For the biggest wave amplitudes tested (x1.0), cBathy gets better results offshore, while uBathy is capable to get also some good results near the coast (Table 4.3, cases with structures in the bathymetry). Then, these inversion algorithms are capable to obtain acceptable results with waves whose amplitude is at most near 15 cm for a polychromatic wave field (the sum of the composing monochromatic waves amplitudes in the x0.4 cases) and at least 25 cm for a monochromatic wave field. The complexity of the wave field is indeed a determining factor for the amplitudes involved.

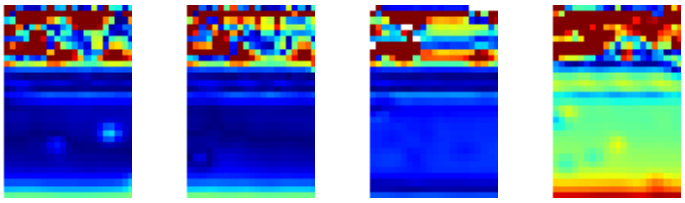
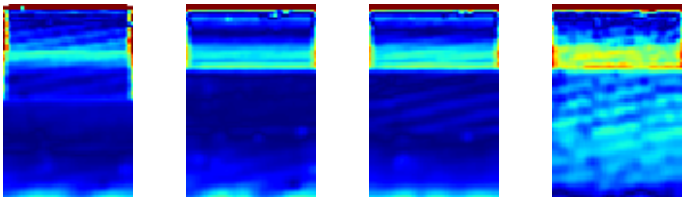
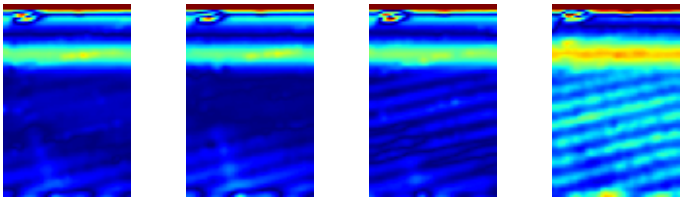
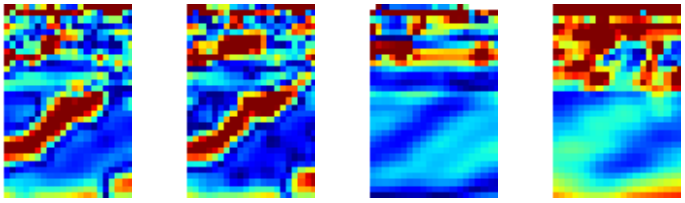
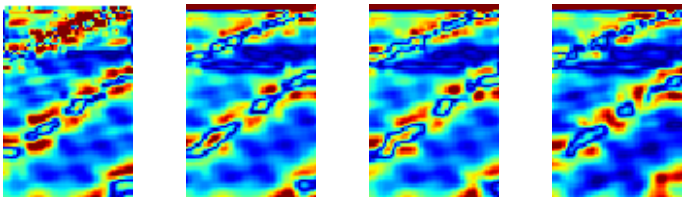
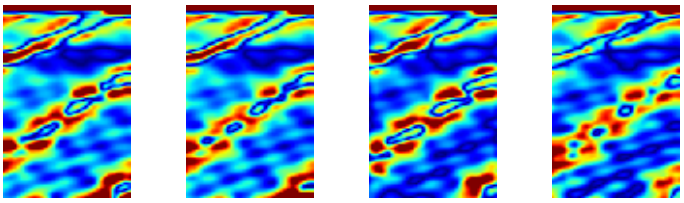
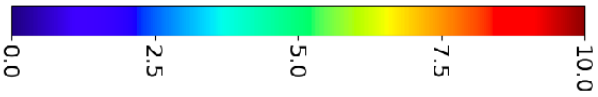
case\sampling frequency	4.00 Hz	2.00 Hz	1.00 Hz	0.50 Hz
linear, baseline, monochromatic, 0.10 ppm				
linear, baseline, monochromatic, 0.25 ppm				
linear, baseline, monochromatic, 0.50 ppm				
linear, baseline, polychromatic, 0.10 ppm				
linear, baseline, polychromatic, 0.25 ppm				
linear, baseline, polychromatic, 0.50 ppm				
relative error colorbar [%]				

Table 4.1: Effects of the temporal resolution over depth error [%]. cBathy results. Legend: m - monochromatic, p - polychromatic, b - baseline, s - structures, l - linear, f - FUNWAVE.

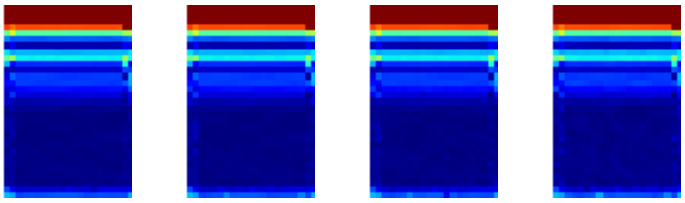
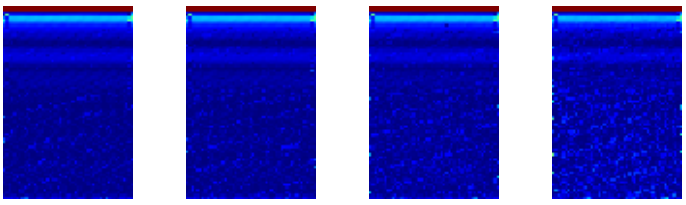
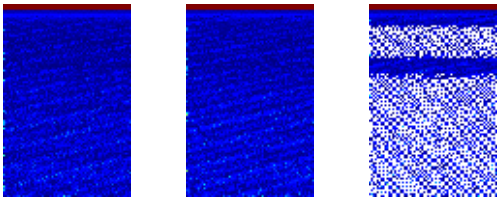
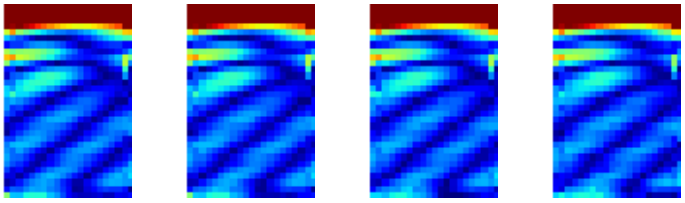
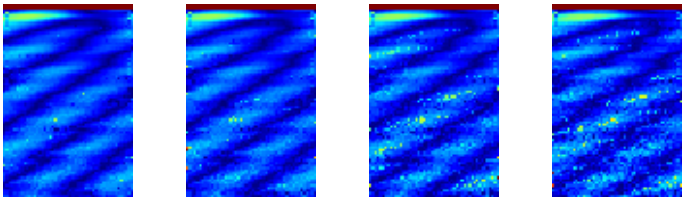
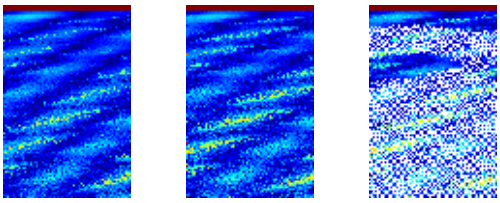
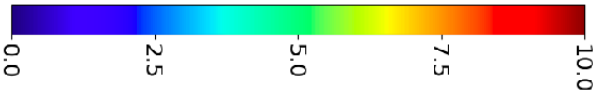
case\sampling frequency	4.00 Hz	2.00 Hz	1.00 Hz	0.50 Hz
linear, baseline, monochromatic, 0.10 ppm				
linear, baseline, monochromatic, 0.25 ppm				
linear, baseline, monochromatic, 0.50 ppm				x
linear, baseline, polychromatic, 0.10 ppm				
linear, baseline, polychromatic, 0.25 ppm				
linear, baseline, polychromatic, 0.50 ppm				x
relative error colorbar [%]				

Table 4.2: Effects of the temporal resolution over depth error [%]. uBathy results. Legend: m - monochromatic, p - polychromatic, b - baseline, s - structures, l - linear, f - FUNWAVE.

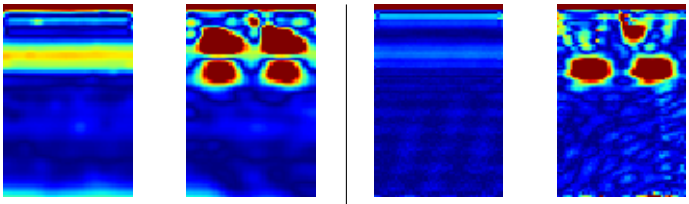
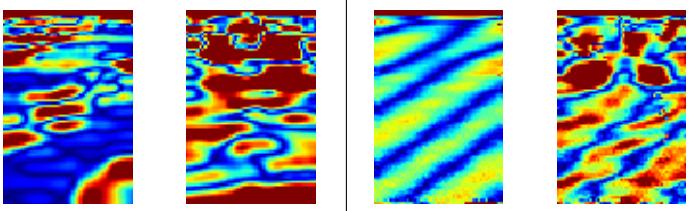
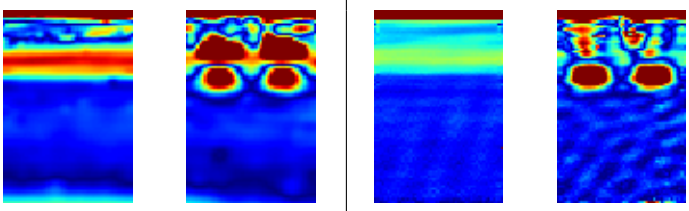
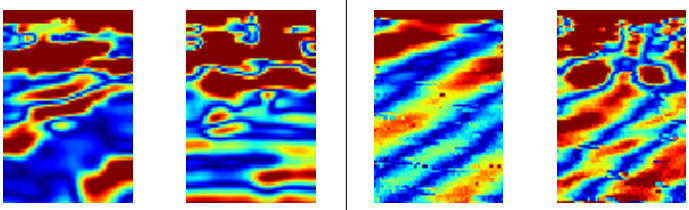
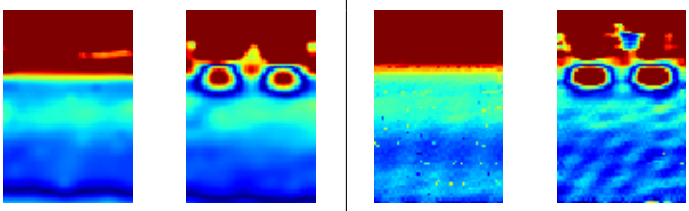
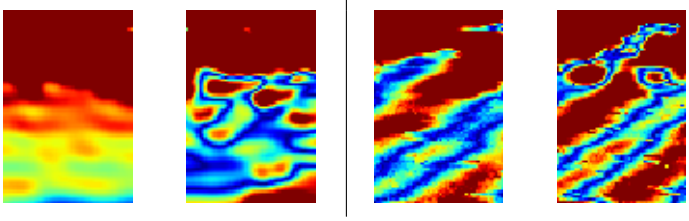
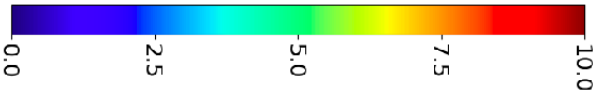
case\bathymetry	cBathy relative error [%] baseline	cBathy relative error [%] structures	uBathy relative error [%] baseline	uBathy relative error [%] structures
funwave, monochromatic, 0.5 Hz, 0.25 ppm, x0.1				
funwave, polychromatic, 0.5 Hz, 0.25 ppm, x0.1				
funwave, monochromatic, 0.5 Hz, 0.25 ppm, x0.4				
funwave, polychromatic, 0.5 Hz, 0.25 ppm, x0.4				
funwave, monochromatic, 0.5 Hz, 0.25 ppm, x1.0				
funwave, polychromatic, 0.5 Hz, 0.25 ppm, x1.0				
relative error colorbar [%]				

Table 4.3: Effects of the bathymetry type over depth error [%]. cBathy results. Legend: m - monochromatic, p - polychromatic, b - baseline, s - structures, l - linear, f - FUNWAVE.

4.3 Discussion

In general, uBathy presents better results than cBathy, specially for the linear cases (it is quite noticeable for the linear polychromatic ones). CBathy seems to get stronger but more localized error zones, while uBathy results present milder but more outspread errors. Another interesting fact is that the errors they present use to mimic the shape of the waves used for the inversion, specially for the polychromatic cases (see the bottom rows of Tables 4.1 and 4.2, corresponding to polychromatic waves).

If uBathy and the modified cBathy were to be used in real conditions, the complexity of the incoming waves (this is, if the inversion case is monochromatic or polychromatic) and the amplitude of those waves will limit the cases where they can be used. The inversion can be skipped if the case is a monochromatic wave of great amplitude, or if the case is a polychromatic wave with composing amplitudes bigger than 15 cm (if the waves are monochromatic or not is easy to check using the video images, and the amplitude can be obtained from a buoy placed near the inversion domain). A great improvement would be to always perform the inversions, but adding a Kalman filter to mix their results. The information about the waves type and amplitude can be used in the weights used by the filter. Moreover, uBathy can be used in the cases of smaller amplitude, and cBathy in the cases of bigger ones, in order to obtain the best results.

Finally, in order to quantify the general distribution of errors present in the inversions, the quartiles and the mean of pixels depth absolute error have been computed and plotted (Table 4.4 and Figure 4.3). The plots for the case of relative errors are also shown (Figure 4.2). The RMS_{95} can be used to compare the results obtained in this work with those from previous studies. The RMS_{95} obtained by uBathy in the linear cases is almost 10 cm. The RMS_{95} obtained by cBathy in the linear cases is between 10 cm and 20 cm approximately. For the cases with non-infinitesimal amplitude, the RMS_{95} errors depend strongly on the amplitude and the complexity of the incoming waves and on the bathymetry type, presenting a great variability. For both cBathy and uBathy, the RMS_{95} is somewhere between 10 cm and 40 cm. These errors are of the same order than those obtained by Holman et al. [2013] (RMS of 51 cm and 56 cm, original cBathy) and Rutten et al. [2017] (RMS between 34 cm and 1 m depending on depth, original cBathy), generally somehow smaller. It has to be noted that their studies used real data, but they applied a Kalman filter to the single bathymetry estimates. The errors stated correspond to those present in the filtered bathymetries. Taking into account the degree of improvement that the modified cBathy single video inversions present over the original cBathy (Figure 3.5), and also that the Kalman filter produces bathymetries whose errors are quite smaller than those coming from a single video, seems plausible to think that since our RMS errors

and those presented in Holman et al. [2013] and Rutten et al. [2017] are quite similar, the methods presented in this work (modified cBathy and uBathy) provide a significant improvement to the previous existing methods.

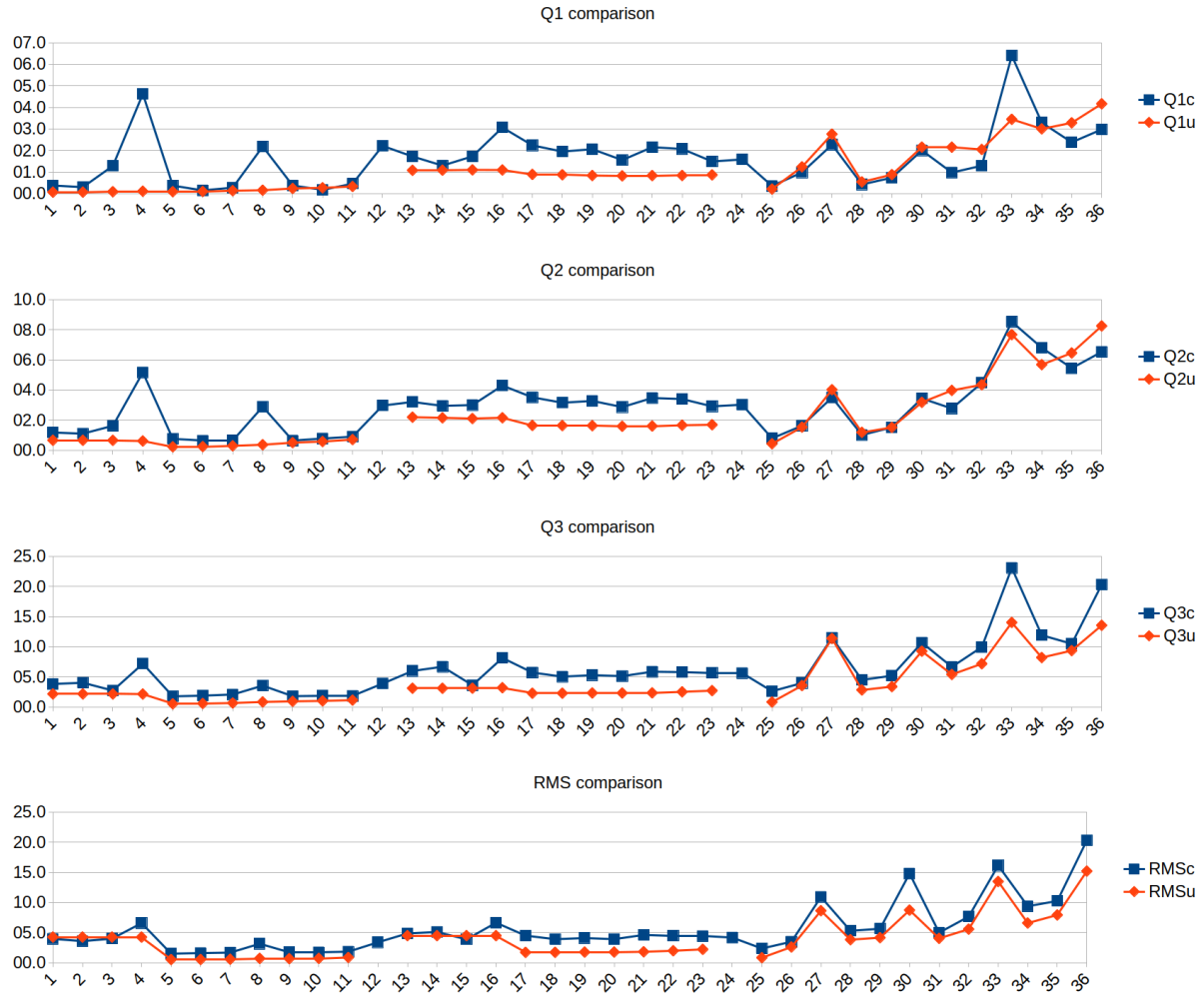


Figure 4.2: Comparison of the quartiles and RMS of pixels relative depth error [%] between the two inversion methods.

No.	Case description	cBathy				uBathy			
		Q1 [cm]	Q2 [cm]	Q3 [cm]	RMS [cm]	Q1 [cm]	Q2 [cm]	Q3 [cm]	RMS_{95} [cm]
01	lbm, 4.0 Hz, 0.10 ppm	01.8	04.9	13.3	10.9	00.5	02.1	07.4	07.5
02	lbm, 2.0 Hz, 0.10 ppm	01.5	04.8	12.1	10.4	00.5	02.1	07.4	07.5
03	lbm, 1.0 Hz, 0.10 ppm	05.0	09.0	11.3	10.1	00.5	02.1	07.4	07.5
04	lbm, 0.5 Hz, 0.10 ppm	17.1	25.0	33.7	27.6	00.7	02.2	07.3	07.5
05	lbm, 4.0 Hz, 0.25 ppm	01.6	03.1	06.2	05.2	00.4	01.0	01.7	01.6
06	lbm, 2.0 Hz, 0.25 ppm	00.7	02.6	07.2	05.4	00.5	01.1	01.8	01.7
07	lbm, 1.0 Hz, 0.25 ppm	01.3	03.0	06.8	05.5	00.6	01.3	02.3	02.0
08	lbm, 0.5 Hz, 0.25 ppm	08.6	13.2	18.0	13.6	00.7	01.5	03.2	02.9
09	lbm, 4.0 Hz, 0.50 ppm	01.8	03.1	06.3	05.0	00.8	01.9	04.7	03.7
10	lbm, 2.0 Hz, 0.50 ppm	00.8	03.4	07.3	05.5	00.9	02.2	05.2	03.9
11	lbm, 1.0 Hz, 0.50 ppm	02.2	04.2	06.8	05.4	00.9	02.2	05.5	04.6
12	lbm, 0.5 Hz, 0.50 ppm	10.2	13.5	17.4	14.0	xx.x	xx.x	xx.x	xx.x
13	lbp, 4.0 Hz, 0.10 ppm	06.8	12.0	21.6	20.3	04.7	10.0	15.5	11.6
14	lbp, 2.0 Hz, 0.10 ppm	05.4	10.8	23.4	20.1	04.8	09.8	15.5	11.5
15	lbp, 1.0 Hz, 0.10 ppm	07.2	13.4	19.0	14.1	04.7	09.8	15.2	11.5
16	lbp, 0.5 Hz, 0.10 ppm	14.1	20.3	27.3	23.1	05.0	09.7	15.0	11.5
17	lbp, 4.0 Hz, 0.25 ppm	08.5	14.3	24.1	18.9	03.0	06.3	11.4	08.4
18	lbp, 2.0 Hz, 0.25 ppm	07.6	12.8	20.7	16.9	02.9	06.3	11.4	08.4
19	lbp, 1.0 Hz, 0.25 ppm	08.0	13.4	21.4	17.7	02.9	06.1	11.4	08.5
20	lbp, 0.5 Hz, 0.25 ppm	05.9	11.4	20.0	16.7	02.8	06.2	11.3	08.6
21	lbp, 4.0 Hz, 0.50 ppm	08.4	14.3	23.8	20.0	02.7	05.8	11.9	09.2
22	lbp, 2.0 Hz, 0.50 ppm	08.1	14.0	23.3	19.3	02.8	06.1	12.7	10.1
23	lbp, 1.0 Hz, 0.50 ppm	05.8	11.6	22.0	18.4	02.5	05.4	12.6	10.8
24	lbp, 0.5 Hz, 0.50 ppm	05.9	11.8	22.0	18.4	xx.x	xx.x	xx.x	xx.x
25	fbm, 2.0 Hz, 0.25 ppm, x0.1	01.7	03.9	08.1	06.8	01.0	02.0	03.4	02.5
26	fbm, 2.0 Hz, 0.25 ppm, x0.4	04.7	07.6	12.0	09.9	06.3	07.8	09.4	08.0
27	fbm, 2.0 Hz, 0.25 ppm, x1.0	12.9	17.6	26.4	24.0	15.9	19.4	25.0	21.4
28	fsm, 2.0 Hz, 0.25 ppm, x0.1	02.2	04.7	11.6	12.0	02.5	05.3	10.3	10.9
29	fsm, 2.0 Hz, 0.25 ppm, x0.4	04.0	07.3	12.5	12.2	04.2	07.3	11.2	10.1
30	fsm, 2.0 Hz, 0.25 ppm, x1.0	12.2	17.2	26.1	26.4	12.3	16.5	22.4	21.0
31	fbp, 2.0 Hz, 0.25 ppm, x0.1	04.3	09.0	21.4	19.2	06.8	14.4	26.6	19.6
32	fbp, 2.0 Hz, 0.25 ppm, x0.4	06.4	14.9	33.8	25.0	08.5	18.5	29.2	21.8
33	fbp, 2.0 Hz, 0.25 ppm, x1.0	38.3	42.8	53.1	45.5	16.7	34.8	52.9	39.9
34	fsp, 2.0 Hz, 0.25 ppm, x0.1	11.5	26.7	46.2	35.8	10.2	22.9	40.1	29.1
35	fsp, 2.0 Hz, 0.25 ppm, x0.4	10.5	22.4	38.7	29.5	12.1	25.4	42.2	31.7
36	fsp, 2.0 Hz, 0.25 ppm, x1.0	16.6	31.6	53.8	41.9	18.7	38.8	61.1	46.3

Table 4.4: Metrics for the absolute error distribution, in cm. Legend: m - monochromatic, p - polychromatic, b - baseline, s - structures, l - linear, f - FUNWAVE, No. - case number used to plot each case in the graphs shown below.

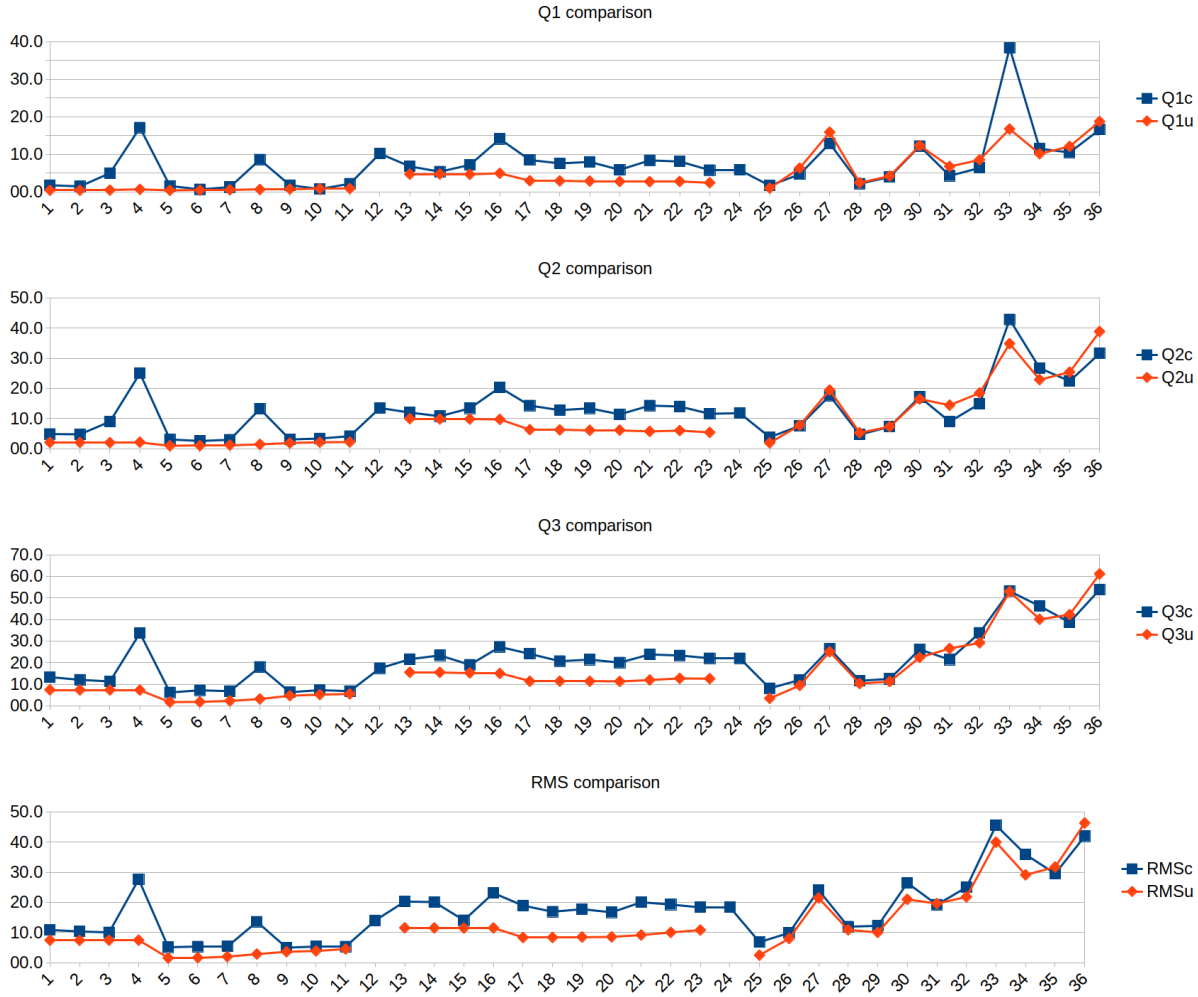


Figure 4.3: Comparison of the quartiles and RMS of pixels absolute depth error [cm] between the two inversion methods.

5 . Conclusions, Outcomes and Future Work

Simulating the propagation of waves over artificial bathymetries has proven to be effective for studying inversion algorithms in a controlled way. It has been specially useful for determining how they respond to the variation of the input video parameters. An environment for obtaining videos of synthetic waves simulating of the desired characteristics has been developed, based on FUNWAVE and linear wave solvers.

The popular inversion algorithm cBathy has been analyzed and adapted. In the process, it has been found that the selection of frequential and spatial parameters it uses has critical effect on the results obtained. The dependence with the frequential parameters has been outlined. Accordingly, a routine to select the best frequential set of parameters has been developed, and another one has been proposed for the spatial parameters.

A new inversion algorithm called uBathy has been presented and compared with cBathy, using a reduced set of study cases. It has been observed that uBathy tends to perform better and to provide more stable results than cBathy, at least for the cases studied.

The best temporal and spatial resolutions for the input videos have been determined (about 2 Hz and 0.25 ppm), supporting the values suggested by previous studies. Also, the range of amplitudes useful for inversion has been found (up to 15 cm for a polychromatic case, and up to 10 cm in a monochromatic case). The spatial distribution of errors has shown that the errors associated with cBathy inversions are stronger but more localized, while the uBathy ones are feebler but also more spread.

However, there is still work to be done. Regarding the video generation process, to extend the colouring algorithm to mimic the images obtained by a camera, by modelling how the light interacts with the waves. About cBathy modifications, to improve the tuning of parameters in cBathy, by implementing a continuous coherence, finding better metrics and adding a peak detection algorithm. Once each inversion algorithms are perfected, to develop a Kalman filter in order to fuse the results of different inversions. Finally, to use the video generation process and the inversion algorithms to study how the morphological features of bathymetries affect the inversion process.

References

L.D Wright and A.D Short. Morphodynamic variability of surf zones and beaches: A synthesis. *Marine Geology*, 56(1-4):93–118, 1984. doi: 10.1016/0025-3227(84)90008-2.

Giovanni Coco, Daniel Calvete, Albert Falqués, and Nicholas Dodd. The role of initial bathymetry on rip channel formation. In *River, Coastal and Estuarine Morphodynamics: RCEM 2007, Two Volume Set*, pages 745–751. CRC Press, 2007. doi: 10.1201/noe0415453639-c95.

J.P. Dugan, W.D. Morris, K.C. Vierra, C.C. Piotrowski, G.J. Farruggia, and D.C. Campion. Jetski-based nearshore bathymetric and current survey system. *Journal of Coastal Research*, 17:900–908, 09 2001.

William A. Birkemeier and Curtis Mason. The crab: A unique nearshore surveying vehicle. *Journal of Surveying Engineering*, 110(1):1–7, 1984. doi: 10.1061/(asce)0733-9453(1984)110:1(1).

Jennifer L. Irish and W.Jeff Lillycrop. Scanning laser mapping of the coastal zone: the SHOALS system. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2-3): 123–129, 1999. doi: 10.1016/s0924-2716(99)00003-9.

D.R. Lyzenga, N.P. Malinas, and F.J. Tanis. Multispectral bathymetry using a simple physically based algorithm. *IEEE Transactions on Geoscience and Remote Sensing*, 44 (8):2251–2259, 2006. doi: 10.1109/tgrs.2006.872909.

W.W. Williams. The determination of gradients on enemy-held beaches. *The Geographical Journal*, 109(1/3):76–90, 1947.

T. C. Lippmann and R. A. Holman. Quantification of sand bar morphology: A video technique based on wave dissipation. *Journal of Geophysical Research*, 94(C1):995, 1989. doi: 10.1029/jc094ic01p00995.

Stéphan T Grilli. Depth inversion in shallow water based on nonlinear properties of shoaling periodic waves. *Coastal Engineering*, 35(3):185–209, 1998. doi: 10.1016/s0378-3839(98)00035-0.

Ap van Dongeren, Nathaniel Plant, Anna Cohen, Dano Roelvink, Merrick C. Haller, and Patricio Catalán. Beach wizard: Nearshore bathymetry estimation through assimilation of model computations and remote observations. *Coastal Engineering*, 55(12):1016–1027, 2008. doi: 10.1016/j.coastaleng.2008.04.011.

Patricio A. Catálan and Merrick C. Haller. Remote sensing of breaking wave phase speeds

with application to non-linear depth inversions. *Coastal Engineering*, 55(1):93–111, 2008. doi: 10.1016/j.coastaleng.2007.09.010.

Stylianos Flampouris, Joerg Seemann, Christian Senet, and Friedwart Ziemer. The influence of the inverted sea wave theories on the derivation of coastal bathymetry. *IEEE Geoscience and Remote Sensing Letters*, 8(3):436–440, 2011. doi: 10.1109/lgrs.2010.2082491.

D.B. Trizna. Errors in bathymetric retrievals using linear dispersion in 3-d FFT analysis of marine radar ocean wave imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 39(11):2465–2469, 2001. doi: 10.1109/36.964983.

C.M. Senet, J. Seemann, S. Flampouris, and F. Ziemer. Determination of bathymetric and current maps by the method DiSC based on the analysis of nautical x-band radar image sequences of the sea surface (november 2007). *IEEE Transactions on Geoscience and Remote Sensing*, 46(8):2267–2279, 2008. doi: 10.1109/tgrs.2008.916474.

R. Holman, N. Plant, and T. Holland. cbathy: A robust algorithm for estimating nearshore bathymetry. *Journal of Geophysical Research: Oceans*, 118(5):2595–2609, 2013. doi: 10.1002/jgrc.20199.

Rob Holman and Merrick C. Haller. Remote sensing of the nearshore. *Annual Review of Marine Science*, 5(1):95–113, 2013. doi: 10.1146/annurev-marine-121211-172408.

Jantien Rutten, Steven M. de Jong, and Gerben Ruessink. Accuracy of nearshore bathymetry inverted from ${x}$ -band radar and optical video data. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2):1106–1116, 2017. doi: 10.1109/tgrs.2016.2619481.

D.H. Peregrine. Equations for water waves and the approximation behind them. In R Meyer, editor, *Waves on Beaches*, pages 95–121. Academic Press, 1972.

O. Nwogu. Alternative form of boussinesq equations for nearshore wave propagation. *J. Waterw., Port, Coastal, Ocean Eng.*, 119(6):618–638, 1993.

Q. Chen. Fully nonlinear boussinesq-type equations for waves and currents over porous beds. *J. Eng. Mech.*, 132(2):220–230, 2006.

A.B. Kennedy, J.T. Kirby, and R.A. Dalrymple. Boussinesq-type equations with improved nonlinear performance. *Wave Motion*, 33:225–243, 2001.

Gonzalo Simarro, Alejandro Orfila, and Alvaro Galan. Linear shoaling in boussinesq-type wave propagation models. *Coastal Engineering*, 80:100–106, 2013. doi: 10.1016/j.coastaleng.2013.05.009.

G. Wei, J.T. Kirby, and A. Sinha. Generation of waves in boussinesq models using a source function method. *Coastal Engineering*, 36:271–299, 1999.

G. Simarro. Energy balance, wave shoaling and celerity in boussinesq-type wave propagation models. *Ocean Model*, 72:74–79, 2013.

J. Yu and D.N. Slinn. Effect of wave-current interaction on rip currents. *J. Geophys. Res.*, 108(C3), 2003. doi: 10.1029/2001JC001105.