Chapter 5

OptiMorph 2.0 User Guide

In this user guide, we present the new version of the OptiMorph model based on the coastal hydro-morphodynamics by minimization principle. This morphodynamic model can be coupled with any hydrodynamic model. We therefore present how to couple this model with hydrodynamic models such as XBeach or SWAN.

Current chapter contents

1 Introduction . . . **135**
1.1 About . . . 135
1.2 Expectation and Objectives . . . 135
1.3 Target Audience . . . 135
2 Processes and Theoretical Formulation . . . **136**
2.1 Domain and Definitions . . . 136
2.1.1 Tide . . . 136
2.2 Hydrodynamic Models . . . 137
2.2.1 Extended Shoaling Model . . . 137
2.2.2 XBeach Model . . . 139
2.2.3 SWAN Model . . . 141
2.3 Morphodynamic Model by Wave Energy Minimization . . . 142
2.3.1 Introduction . . . 143
2.3.2 Governing Equation of Seabed Dynamics . . . 143
2.3.3 Parameter Υ . . . 143
2.3.4 Parameter Λ . . . 144
2.3.5 Direction of Descent d . . . 145
2.3.6 Choice of Cost Function $\mathcal{J}$ . . . 146
2.3.7 Hadamard Derivative to Compute $\nabla_\psi \mathcal{J}$ . . . 146
2.3.8 Slope Limiter . . . 147
2.4 Model Constraints . . . 149
2.4.1 Slope Constraint . . . 149
2.4.2 Sand Stock Constraint . . . 150
3 Numerical Model . . . **151**
3.1 Presentation . . . 151
3.1.1 Workflow . . . 151
3.1.2 Program Organization . . . 152
3.2 Running OptiMorph . . . 155

3.2.1 Installation of OptiMorph 155
3.2.2 Installation of PAGURE (to install SWAN and XBEACH) . . 156
3.2.3 Input File 165
4 Applications 167
4.1 1D Linear Seabed Beach Configuration using Hadamard approach with SWAN 168
4.1.1 Setting 168
4.1.2 Input files 168
4.1.3 Load SWAN and Run OptiMorph on Cluster 170
4.1.4 Results 170
4.2 1D Linear Seabed Beach with GeoTube using Hadamard approach with XBeach 173
4.2.1 Setting 173
4.2.2 Input Files 173
4.2.3 Load XBeach and Run OptiMorph on Cluster 175
4.2.4 Results 175
4.3 2D Linear Seabed Beach Configuration using Hadamard approach with Shoaling 178
4.3.1 Setting 178
4.3.2 Input Files 178
4.3.3 Results 180

1 Introduction

1.1 About

The numerical hydro-morphodynamic model presented here embodies a new approach to coastal morphodynamics, based on optimization theory. The model we present here follows on from the model presented by Cook et al. (2021a) in his user guide. This one is also based on the assumption that a sandy seabed evolves over time in order to minimize a certain wave-related function, the choice of which depends on what is considered the driving force behind coastal morphodynamics. This numerical model was given the name OptiMorph, and has the advantages of being fast, robust, and requires very few input parameters.

The model we present in this user guide is based on the same principles as Cook et al. (2021a) model. However, this model has been recreated with new enhancements, which we present in this user guide.

1.2 Expectation and Objectives

The main goal of OptiMorph was to demonstrate the potential of using optimal control in the modeling of coastal dynamics by designing an adaptable, easy-to-use numerical model. Our model aims to be generic, fast, robust and easy to use. It is intended to act as a morphodynamic module that can be coupled with any hydrodynamic model. This model aims to simulate morphodynamic phenomenology (creation of sedimentary bars) very well. It could eventually be incorporated into morphodynamic models without this phenomenological aspect.

1.3 Target Audience

The OptiMorph model is a tool intended for any person wishing to simulate the natural evolution of the coastal seabed in response to the incoming wave conditions, and/or to study the effect of man-made submerged devices on the sediment transport. It can be used by engineers seeking an opinion on the morphodynamic aspect of a project. By students wishing to understand morphodynamic phenomena. By a morphodynamic developer looking to improve his model...

2 Processes and Theoretical Formulation

2.1 Domain and Definitions

We consider a coordinate system composed of a horizontal axis x and a vertical axis z. We denote $\Omega := [0, x_{\max}]$ the domain of the cross-shore profile of the active coastal zone, where $x = 0$ is a fixed point in deep water where no significant change in bottom elevation can occur, and $x_{\max}$ is an arbitrary point at the shore beyond the shoreline, as shown by Figure 5.1. The elevation of the sea bottom is a one-dimensional positive function, defined by: $\psi : \Omega \times [0, T_f] \times \Psi \to \mathbb{R}^+$ where $[0, T_f]$ is the duration of the simulation (s) and Ψ is the set of physical parameters describing the characteristics of the beach profile. In order to model the evolution over time of ψ and given the assumption that ψ changes over time in response to the energy of shoaling waves, a description of the surface waves is needed.

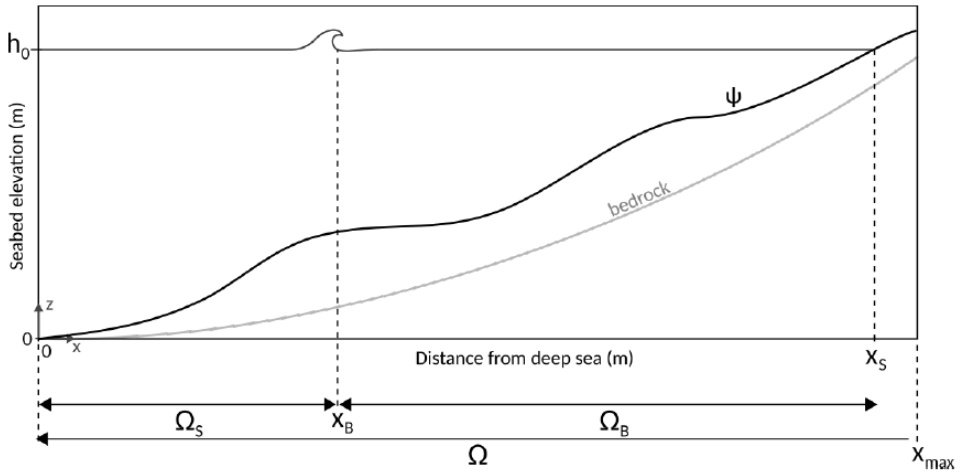


Figure 5.1 – Illustration of the cross-shore profile where breaking occurs once at $x = x_B$.

ⓘ Here, we present only part 1D. For the 2D part, the notations are analogous.

2.1.1 Tide

In this model, some parameters are time-variable: $T_0(t)$, $h_0(t)$, $H_0(t)$,... The choice of a temporally variable closure depth $h_0(t)$ allows to manage the effect of tides. This is defined as follows:

$$h_0(t) = \frac{M_{eff}}{2} \sin\left(\frac{2\pi t}{T_{tide}}\right) \quad with \quad Ma = \frac{C_{tide} M_{ref}}{100} \tag{5.1}$$

with M_{ref} the reference tidal range (m): for example, in Brest, it is 6.1 m. M_{eff} the effective tidal range (m), C_{tide} the tidal coefficient and T_{tide} the tide duration (s).

2.2 Hydrodynamic Models

Unlike the previous user guide of Optimorph (Cook et al. 2021a), this one will focus solely on 3 hydrodynamic models. Indeed, numerical advances have enabled us to couple hydrodynamic models that are well known in the literature. We no longer need to develop models ourselves. The first is a purely numerical model based on the W. Munk (1949) criterion: this model enables the code to be tested very quickly. The other two models are XBeach (D. J. Roelvink et al. 2009; Zimmermann et al. 2012; Bugajny et al. 2013; Williams et al. 2015) and SWAN (Booij et al. 1996).

2.2.1 Extended Shoaling Model

The Shoaling model (Cook 2021) did not succeed to model wave breaking with wave periods $T_0 > 2\,s$. This model was therefore improved to give birth to the extended model below:

Extended Shoaling model

$$H(x,t) = \begin{cases} H_0(x,t)K_S(x,t) & \text{for } x \in \Omega_S \quad (5.2a) \\ \mathcal{F}\left(\gamma h(x,t)\right) & \text{for } x \in \Omega_B \quad (5.2b) \end{cases}$$

where $\mathcal{F}$ is a numerical parameterization function of the breaking define below (5.3):

$$\mathcal{F}\left(\gamma h(x,t)\right) = H(x_{start}) + \left[H(x_{stop}) - H(x_{start})\right] \cdot f\left(\frac{x - x_{start}}{x_{stop} - x_{start}}\right) \cdot g\left(\frac{h_{max} - h}{h_{max} - h_{min}}\right) \quad (5.3)$$

with $x \in \Omega_B = [x_{start}, x_{stop}]$, $h \in [h_{min}, h_{max}]$ and the following notations:

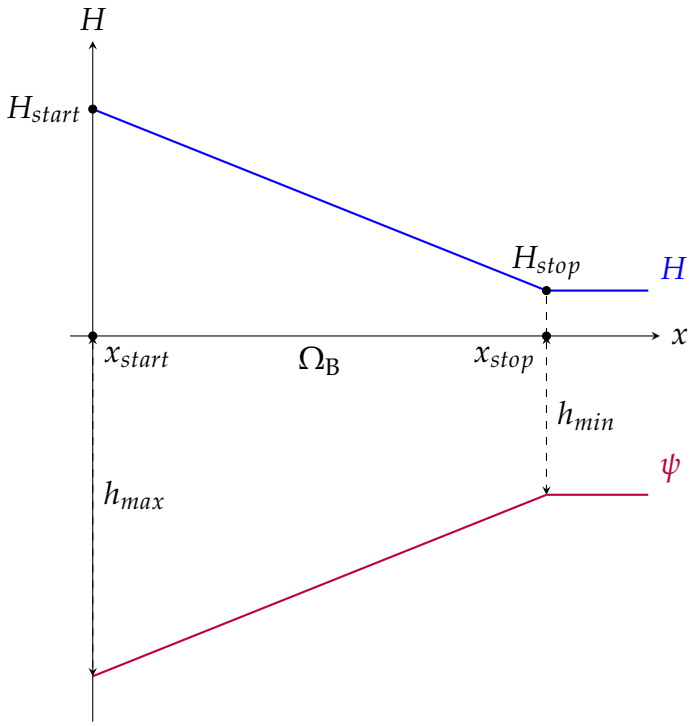


Figure 5.2 – Illustration of notations.

H_{start} and H_{stop} are the water heights at the beginning and end of the breaking on the domain $\Omega_B = [x_{start}, x_{stop}]$. The first function f gives an account of the breaking without taking into account the bed shape. It simply gives the appearance of the breaking. The second function g takes into account the seabed and interacts with it. Note that if f and g are the affine functions $x \longmapsto x$, we find the breaking $\gamma h(x,t)$ illustrated on figure 5.2. We can present below (figure 5.3) some of these functions that set the breaking:

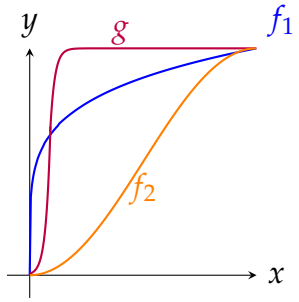


Figure 5.3 – Illustration of f_1, f_2 and g defined in $[0,1] \longrightarrow [0,1]$.

These functions were chosen to try to capture a natural breaking. They have no physical meaning.

It is necessary to stipulate that the model will first locate all the Ω_B domains and then apply the equation (5.3) on each of them. The result of this model can be seen with the breaker shown in figure 5.29.

2.2.2 XBeach Model

The XBeach model is a process-based model developed by the Delft University of Technology. It is a two-dimensional, depth-integrated numerical model that simulates the hydrodynamics, sediment transport, and morphological changes of coastal systems. XBeach is a flexible model that can be used to simulate a variety of coastal processes, including wave breaking, bedload transport, and nearshore morphological changes. The model is based on the principles of conservation of mass, momentum, and energy and uses a finite-difference numerical scheme to solve the governing equations. XBeach has been widely used in coastal studies due to its flexibility and accuracy, and it has been applied to a wide range of coastal systems, including estuaries, beaches, and coastal wetlands. The model can be used as a profile model in 1D (Pender et al. 2013), or as an area model in 2D (McCall et al. 2010), and today, there are three modes in which the hydrodynamics can be resolved in XBeach, being:

- **Stationary** – All wave group variations, and thereby all infragravity motions, are neglected, and only the mean motions are included. This type can be applied for modeling morphological changes under moderate wave conditions;
- **Surfbeat** – This in-stationary, hydrostatic mode, is wave group resolving, and is hence also applicable when one is interested in the swash zone processes;
- **Non-hydrostatic** – The non-linear shallow water equations are solved, and hence individual short wave propagation and transformation is resolved.

In our case, we will focus on the **Stationary** mode.

2.2.2.1 Hydrodynamics

The wave action balance is solved to obtain the wave forcing:

$$\frac{\partial A}{\partial t} + \frac{\partial c_x A}{\partial x} + \frac{\partial c_y A}{\partial y} + \frac{\partial c_\theta A}{\partial \theta} = -\frac{D_w}{\sigma} \tag{5.4}$$

Where A is the wave action, C the wave propagation speed (where the subscripts refer to the $x-$ and $y-$directions, and $\theta-$space), θ is the angle of incidence, D_w the wave energy dissipation per directional bin and σ the intrinsic wave frequency. The wave action as above (5.5) by:

$$A(x, y, t, \theta) = \frac{S_w(x, y, t, \theta)}{\sigma(x, y, t)} \tag{5.5}$$

In which the Sw is the wave energy density per directional bin. The total wave energy E_H is obtained by integration of the wave energy density Sw over all directional bins:

$$E_H = \int_0^{2\pi} S_w(x, y, t, \theta) \mathrm{d}\theta \tag{5.6}$$

The distribution of the total wave energy dissipation $\bar{D}w$ over all directional bins is calculated proportional to the energy density distribution as follows:

$$D_w(x, y, t, \theta) = \frac{S_w(x, y, t, \theta)}{E_w(x, y, t)} \bar{D}_w(x, y, t) \tag{5.7}$$

The total wave energy dissipation is calculated using a method described by D. J. Roelvink (1993) as the product of the dissipation per breaking event and the fraction of broken waves Qb. The energy dissipation per wave breaking event is assumed to take place over half of the representative wave period T_{rep}, resulting in the following expression for the total, directionally integrated, wave energy dissipation:

$$\bar{D}_w = \alpha \frac{2}{T_{rep}} Q_b E_H \tag{5.8}$$

Where α is a calibration factor and E_w the total wave energy (Equation (5.6)). The fraction of breaking waves Qb is estimated from a Rayleigh distribution (Battjes et al. 1978):

$$Q_b = 1 - \exp\left(-\left(\frac{H_{rms}}{H_{\max}}\right)^n\right) \tag{5.9}$$

Where the root-mean-square wave height H_{rms} is calculated from the wave energy E_H, and the maximum wave height H_{max} is calculated using the breaker index γ (the ratio between the breaking wave height and the water depth, usually given the value 0.88).

$$E_H \sim \frac{1}{8}\rho g H_{rms}^2 \quad \Rightarrow \quad H_{rms} = \sqrt{\frac{8E_w}{\rho g}}, \quad H_{\max} = \gamma_b h \tag{5.10}$$

This closes the set of equations for the wave action balance (Equation (5.4)). From the wave energy, the wave-induced radiation stresses can be determined using linear wave theory. Similar to the wave action balance, a roller balance is solved and coupled to the wave energy balance, where the wave energy dissipation forms a source of energy in the roller balance. The roller-induced radiation stress is calculated and together with the wave-induced radiation stress they are used to calculate the wave forcing: The flows are calculated using a depth-averaged formulation of the shallow water equations, taking into account wave-induced mass flux and return flows. This Generalized Lagrangian Mean

(GLM) formulation uses Lagrangian velocities (Andrews et al. 1978):

$$\frac{\partial u^L}{\partial t} + u^L\frac{\partial u^L}{\partial x} + v^L\frac{\partial u^L}{\partial y} - fv^L - v_h\left(\frac{\partial^2 u^L}{\partial x^2} + \frac{\partial^2 u^L}{\partial y^2}\right) = \frac{T_{sx}}{\rho h} - \frac{T^E_{bx}}{\rho h} - g\frac{\partial \eta}{\partial x} + \frac{F_x}{\rho h} \quad (5.11a)$$

$$\frac{\partial v^L}{\partial t} + u^L\frac{\partial v^L}{\partial x} + v^L\frac{\partial v^L}{\partial y} + fu^L - v_h\left(\frac{\partial^2 v^L}{\partial x^2} + \frac{\partial^2 v^L}{\partial y^2}\right) = \frac{T_{sy}}{\rho h} - \frac{T^E_{by}}{\rho h} - g\frac{\partial \eta}{\partial y} + \frac{F_y}{\rho h} \quad (5.11b)$$

$$\frac{\partial \eta}{\partial t} + \frac{\partial u^L h}{\partial x} + \frac{\partial v^L h}{\partial y} = 0 \quad (5.11c)$$

Where the Lagrangian velocity components (denoted by the superscript L) are the superposition of the Eulerian velocity and the Stokes' drift velocity:

$$u^L = u^E + u^S \quad \text{and} \quad v^L = v^E + v^S \quad (5.12)$$

2.2.3 SWAN Model

The SWAN model is a spectral numerical model designed to simulate waves evolving in coastal regions, lakes, and estuaries under defined wind, bathymetry, and current conditions. It is based on the Energy Density Balance equation (**5.5**) linking the advection term to the source and sink terms. Therefore, the wave energy evolves in both geographic and spectral space and changes its aspect due to the presence of wind at the surface, friction with the bottom, or during the breaking of the waves. The SWAN model is a stable model based on unconditionally stable numerical schemes (implicit schemes). SWAN, in its third version, is in stationary mode (optionally non-stationary) and is formulated in Cartesian or spherical coordinates. The unconditional numerical stability of the SWAN model makes its application more effective in shallow water. In SWAN, the waves are described with the two-dimensional spectrum of the wave action density A,

$$A(x, y, \sigma, \theta) = \frac{E(x, y, \sigma, \theta)}{\sigma} \quad (5.13)$$

where x and y are the horizontal components of geographic space, σ is the relative frequency, θ is the wave direction, and E is the energy density.

The spectrum considered in the SWAN model is that of the wave action density $A(\sigma, \theta$) rather than the spectrum of the energy density $E(\sigma, \theta)$. This is because, in the presence of currents for the reasons we mentioned above (non conservation of E_H) (G. B. Whitham 2011). Because wave action propagates in both geographic and spectral space under the influence of genesis and dissipation terms, wave characteristics are described in terms of two-dimensional wave action density. The action density spectrum balance equation relating the propagation term to the effects of the source and sink terms, in Cartesian

coordinates, is (Hasselmann et al. 1973)

$$\frac{\partial A}{\partial t} + \frac{\partial \left(C_x A\right)}{\partial x} + \frac{\partial \left(C_y A\right)}{\partial y} + \frac{\partial \left(C_\sigma A\right)}{\partial \sigma} + \frac{\partial \left(C_\theta A\right)}{\partial \theta} = \frac{S}{\sigma}. \tag{5.14}$$

On the left-hand side of Equation (5.14), the first term represents the local temporal variation of the wave action density, the second and third terms represent the propagation of wave action in the geographical space of velocities C_x and C_y, the fourth term represents the shifting of the relative frequency due to variations in bathymetry (with propagation velocity C_σ) and currents (with propagation velocity C_θ), and the fifth term represents the refraction induced by the combined effects of depth and currents. $C_x, C_y, C_\sigma, C_\theta$ propagation velocities are obtained from linear wave theory. The term in the right-hand side of Equation (5.14) represents processes that generate, dissipate, or redistribute wave energy, and S can be expressed as (Lv et al. 2014)

$$S = S_{in} + S_{wc} + S_{brk} + S_{bot} + S_{n14} + S_{n13} \tag{5.15}$$

where S_{in} is the wind energy input. The dissipation terms of wave energy is represented by the contribution of three terms: the white capping S_{wc}, bottom friction S_{bot}, and depth induced breaking S_{brk}. S_{n14} and S_{n13} represent quadruplet interaction and triad interactions, respectively.

A finite difference scheme is used for each of the five dimensions: time, geographic space, and spectral space made the numerical implementation in SWAN effective. The following guidelines must be followed in order to obtain the discretization adopted at the SWAN model level for numerical computation:

1. time of a constant and identical time step Δt for the propagation term and the source term,
2. geographical space of a rectangular grid with constant spatial steps Δx and Δy,
3. spectral space of a constant directional step $\Delta\theta$ and a constant relative frequency step $\Delta\sigma/\sigma$,
4. frequencies between a fixed minimum maximum values of 0.04 Hz and 1 Hz respectively,
5. the direction θ can also be delimited by the minimum and maximum values θ_{min} and θ_{max} (as an option).

2.3 Morphodynamic Model by Wave Energy Minimization

2.3.1 Introduction

The fundamental assumption governing OptiMorph states that the seabed evolves over time so as to minimize a certain quantity, named cost function. The choice of cost function depends on what is considered the driving force behind the morphodynamic response to the seabed. The one we've chosen calculates wave energy. In other words, the shape of the seabed varies in an effort to minimize the energy of the surface waves at that given time. At each time, the model indicates the direction to a local minimum of the cost function with regard to the parameterization of the seabed. Two physical parameters limit or encourage seabed mobility depending on the proprieties of the sediment and the depth of the water. This optimization problem is subjected to a limited number of constraints, allowing for a more accurate description of the morphodynamic evolution. The first concerns the maximal slope of the seabed, the second manages the sandstock of the profile in the case of an experimental flume.

The optimization problem that OptiMorph seeks to solve is:
For each $t \in [0, T]$, find the shape ψ of the seabed such that the cost function $\mathcal{J}$ is minimal, while subjected to constraints.

The $\mathcal{J}$ calculation is performed using a hydrodynamic model selected from those presented here.

2.3.2 Governing Equation of Seabed Dynamics

The evolution of the sea bottom is assumed to be driven by the minimization of a cost function $\mathcal{J}$ (J s m^{-1}) with the following gradient descent taking the initial sea bottom ψ_0,

$$\begin{cases} \psi_t(.,t) = \Upsilon \, \Lambda \, d(.,t) \\ \psi(.,0) = \psi_0(.). \end{cases} \tag{5.16}$$

where ψ_t is the evolution of the bottom elevation over time (m s^{-1}), Υ is a measure of the sand mobility expressed in m s kg^{-1}, Λ measures the excitation of the seabed by the orbital motion of water waves, and d is the direction of the descent (J s m^{-2}), which indicates the manner in which the sea bottom changes. This approach uses two parameters and two constraints.

2.3.3 Parameter Υ

The first parameter Υ takes into account the physical characteristics of the sand and represents the mobility of the sediment. Simulations with varying Υ that reflect variations of the d_{50} grain diameter from 0.25 mm to 2 mm were performed. Changes in the beach profile were observed but no significant alteration of the trends in beach profile evolution

through time. The asymptotic behavior of the simulations remains the same although the velocity at which a given profile is reached changes. Further explanation of the nature of the Υ parameter will be given at a later stage of the model development. For Υ great, as is the case with finer particles, the seabed may be submitted to significant change. For Υ close to zero, little mobility is observed.

2.3.4 Parameter Λ

The first constraint Υ takes into account the physical characteristics of the sand and represents the mobility of the sediment. The second parameter Λ is a local function which represents the influence of the relative water depth kh on the beach profile dynamics and is defined after the term describing the vertical attenuation of the velocity potential according to linear wave theory (R.L. Soulsby 1987):

$$\begin{aligned} \varphi : \quad \Omega \times [0, h_0] &\longrightarrow \mathbb{R}^+ \\ (x, z) &\longmapsto \frac{\cosh(k(x)(h(x) - (h_0 - z)))}{\cosh(k(x)h(x))} \end{aligned} \tag{5.17}$$

An illustration of the orbital velocity of the wave particles is given in figure 5.4. This function describes the excitation of the water particles for a given location along the cross-shore profile and a given water depth. However, our interest lies in the excitation of the seabed by the surface waves. Therefore, it is natural to consider the orbital damping function at $z = \psi(x)$. The parameter Λ of equation (5.16) is therefore defined by:

$$\Lambda(x) = \varphi(x, \psi(x)) = \frac{1}{\cosh(k(x)h(x))} \tag{5.18}$$

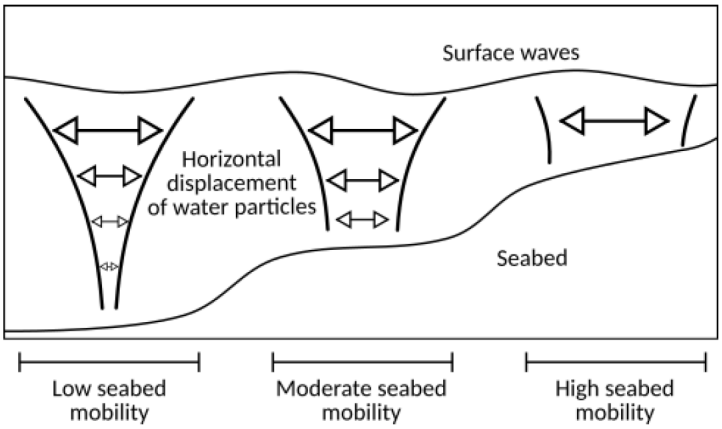


Figure 5.4 – Illustration of the orbital velocity over the cross-shore profile from (Cook 2021)

This parameter governs the manner in which the waves affect the seabed. In deeper waves, the surface waves have little to no effect on the seabed below. No movement should be observed of the seabed, and thus $\Lambda \approx 0$ over this portion of the cross-shore profile. When the waves have a large impact on the seabed, e.g. at the coast, greater movement can be observed and as such we set $\Lambda \approx 1$. An illustration of Λ is given in figure 5.4.

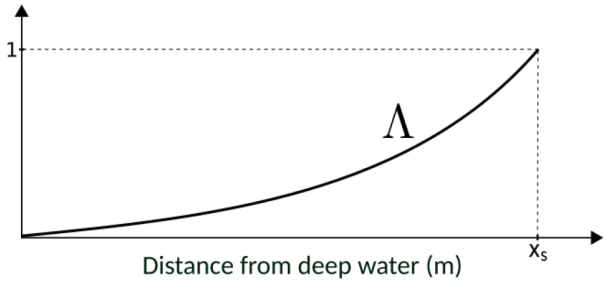


Figure 5.5 – Variation of the parameter Λ over the cross-shore profile from (Cook 2021)

2.3.5 Direction of Descent d

d is the direction of the descent $(\mathrm{J\ s\ m^{-2}})$, which indicates the manner in which the sea bottom changes. In unconstrained configurations, there would be $d = -\nabla_\psi \mathcal{J}$, which by its definition indicates the direction of a local minimum of $\mathcal{J}$ with respect to ψ as illustrated on figure 5.6.

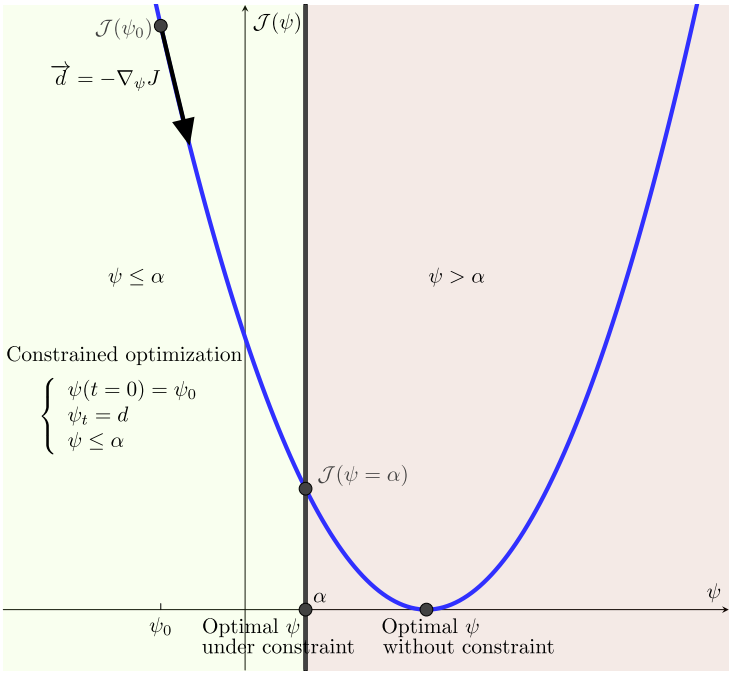


Figure 5.6 – Illustration of gradient descent with $\psi \leq \alpha$. The optimum does not necessarily correspond to the critical point $\nabla_\psi \mathcal{J} = 0$

The tricky step will be to obtain this quantity $\nabla_\psi \mathcal{J}$: this is explained in the section 2.3.7 with Hadamard's derivation.

2.3.6 Choice of Cost Function $\mathcal{J}$

The shape of the beach profile is determined by the minimization of the potential energy of waves, for all $t \in [0, T_f]$:

$$\mathcal{J}(\psi, t) = \frac{1}{16} \int_{t-T_{coupl}}^{t} \int_{\Omega_S} \rho_w g H^2(\psi, x, \tau) \mathrm{d}x \mathrm{d}\tau \tag{5.19}$$

where H denotes the height of the waves over the cross-shore profile (m), ρ_w is water density (kg m^{-3}), and g is the gravitational acceleration (m s^{-2}). T_{coupl} (s) defines the coupling time interval between hydrodynamic and morphodynamic models so that we have T_f / T_{coupl} iterations.

2.3.7 Hadamard Derivative to Compute $\nabla_\psi \mathcal{J}$

We use the approximation described in (Hadamard 1914; Mohammadi 2007; Mohammadi 2010). We consider $\nabla_\psi \mathcal{J}$ in the sense of Hadamard following the definition:

$$\nabla_\psi \mathcal{J} = \lim_{\varepsilon \to 0} \frac{\mathcal{J}(\psi + \varepsilon n) - \mathcal{J}(\psi)}{\varepsilon}, \tag{5.20}$$

where n is the normal to the shape ψ. This can be seen as applying a Gâteaux (1913) derivation in the direction normal to the shape. The principle is illustrated in figure 5.7.

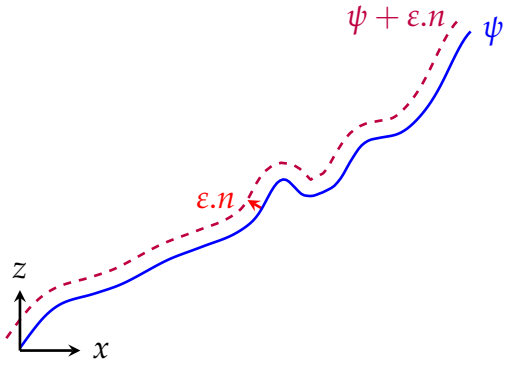


Figure 5.7 – Representation of two sea bottom profiles ψ and $\psi + \varepsilon n$. To calculate the gradient, we need to calculate at all points the associated normal vector n.

Using the Taylor-Young formula at order 1, we consider the following approximation:

$$\begin{aligned}\nabla_\psi \mathcal{J} &= \lim_{\varepsilon \to 0} \frac{\mathcal{J}(\psi) + \varepsilon \nabla_X \mathcal{J}.n - \mathcal{J}(\psi)}{\varepsilon}, \\ &\approx (\nabla_X \mathcal{J}).n,\end{aligned} \tag{5.21}$$

with $X = (x\,,\, z)^\intercal$. To implement this approach practically, we simply need to use the equation (5.21) with: $\nabla_X \mathcal{J} = \begin{pmatrix} \frac{\partial \mathcal{J}}{\partial x} \\ \frac{\partial \mathcal{J}}{\partial \psi} \end{pmatrix}$ and $n = \frac{1}{\sqrt{d\psi^2 + dx^2}} \begin{pmatrix} -d\psi \\ dx \end{pmatrix}$ and we obtain:

$$\nabla_\psi \mathcal{J} \approx \frac{\partial \mathcal{J}}{\partial x} n_x + \frac{\partial \mathcal{J}}{\partial \psi} n_z, \tag{5.22}$$

with n_x and n_z the x and z component of n. $\frac{\partial \mathcal{J}}{\partial x}$ and $\frac{\partial \mathcal{J}}{\partial \psi}$ are calculated using finite differences. The $\Delta\psi$ quantity can sometimes be almost zero, depending on the configuration of the sea bottom. To avoid code explosions, we'll use a slope limiter.

2.3.8 Slope Limiter

A slope limiter was introduced in Hadamard differentiation, and helps maintain stable code. This is based on the following algorithm:

This limiter is applied every time a differentiation is calculated in Hadamard. This limiter is very effective, as shown in the figure 5.8.

Algorithm 1 A slope limiter

Input: y is the vector to limit, n the size of y, n_x the maximum limitation window, often n_x=20

Output: y without degeneration

$error \leftarrow 1$
for Δx=1,n_x **do**
  $y_0 \leftarrow y$
  for i=Δx,$n - \Delta x$ **do**
    $y_{min} \leftarrow \min(y_0[i - \Delta x], y_0[i + \Delta x])$
    $y_{max} \leftarrow \max(y_0[i - \Delta x], y_0[i + \Delta x])$
    $y[i] \leftarrow \max[\min(y_0[i], y_{max}), y_{min}]$
  end for
  $error_0 \leftarrow error$
  $error \leftarrow ||y - y_0||$
  if $\Delta x = 1$ **then**
    $e_0 \leftarrow error$
  end if
  $error \leftarrow \frac{error}{e_0}$
  if $\Delta x > 1$ **and** $error > error0$ **then**
    $y \leftarrow y_0$
    break
  end if
end for

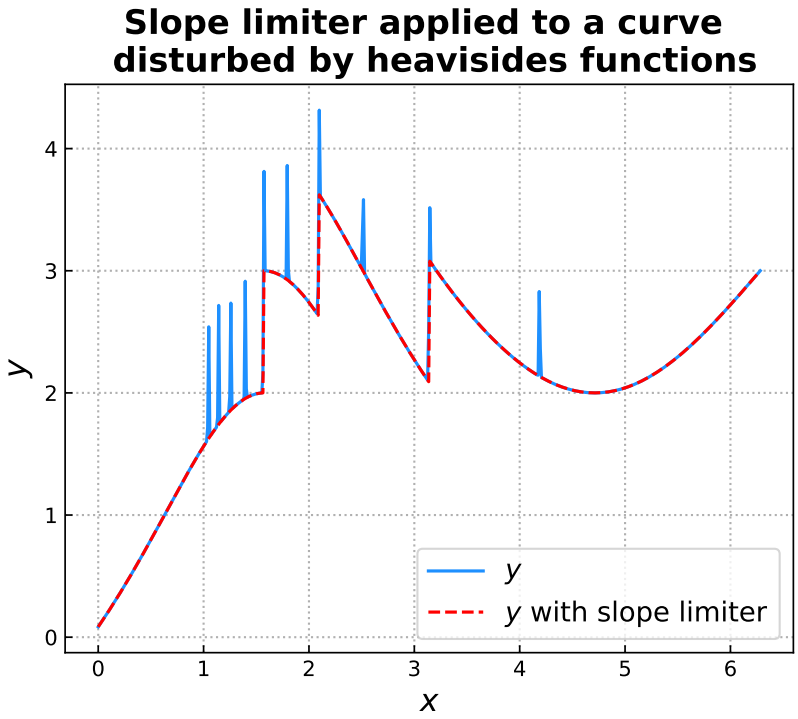


Figure 5.8 – Slope limiter applied to a curve disturbed by heavisides functions

2.4 Model Constraints

In the interest of simplicity, we have adopted two physical constraints though more can be introduced if necessary.

2.4.1 Slope Constraint

The first concerns the local slope of the bottom. Depending on the composition of the sediment, the bottom slope is bounded by a grain-dependent threshold M_{slope}(Dean et al. 2004). This is conveyed by the following constraint on the local bottom slope illustrated by 5.9:

$$\left|\frac{\partial \psi}{\partial x}\right| \leq M_{\text{slope}} \tag{5.23}$$

The dimensionless parameter M_{slope} represents the critical angle of repose of the sediment. This angle is based on observed angles in natural beach environments, which are often between 0.01 and 0.2 (Bascom 1951; Vos et al. 2020; Short 1996). We have considered the observed critical angle of 0.2.

2.4.2 Sand Stock Constraint

A second constraint concerns the sand stock in the case of an experimental flume. In a flume, the quantity of sand must be constant over time, as given by (5.24), contrarily to an open-sea configuration where sand can be transported between the nearshore zone and a domain beyond the closure water depth where sediment is definitely lost for beach morphodynamics (Hattori et al. 1980; Quick 1991). This constraint can be written as :

$$\int_\Omega \psi(t,x)\mathrm{d}x = \int_\Omega \psi_0(x)\mathrm{d}x \qquad \forall t \in [0, T_f] \tag{5.24}$$

This constraint is necessary for verifying and validating the numerical model with the wave flume experimental data.

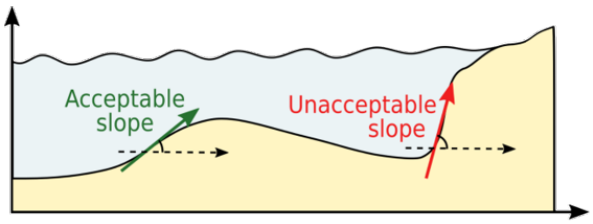


Figure 5.9 – Slope constraint (5.23) from (Cook 2021)

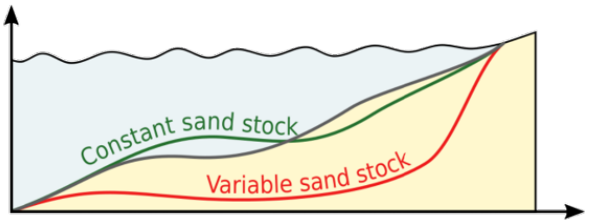


Figure 5.10 – Sand conservation (5.24) from (Cook 2021)

2.4.2.1 Numerical implementation by Gram-Schmidt projections

For the numerical implementation, we introduce a new quantity C_{sand}. For a given time $t \in [0, T_f]$, $C_{\text{sand}}(t)$ is a difference between the current and the initial sandstock, weighted by ψ:

$$C_{\text{sand}}(t) = \int_\Omega (\psi - \psi_0)^2 d\Omega. \tag{5.25}$$

To have a sand conservation (equation (5.24) and figure 5.10), we must have $C_{\text{sand}}(t) = 0$. The minimization problem then becomes:

For each $t \in [0, T]$, find the shape ψ of the seabed such that the cost function $\mathcal{J}$ is minimal, while maintaining $C_{sand}(t) = 0$.

The method chosen to satisfy this constraint is the same as for (Cook 2021), namely the Gram-Schmidt projections.

Since $C_{\text{sand}}(0) = 0$, we wish to minimize $\mathcal{J}$ while keeping C_{sand} constant. This equates to following the direction $\nabla_\psi \mathcal{J}$ while keeping $\nabla_\psi C_{\text{sand}} = 0$. In order to do so, we project the direction $\nabla_\psi \mathcal{J}$ onto the orthogonal of $\nabla_\psi C_{\text{sand}}$. Hence, the direction of

descent d becomes:

$$d = \nabla_\psi \mathcal{J} - \left\langle \nabla_\psi \mathcal{J}, \frac{\nabla_\psi C_{\text{sand}}}{\| \nabla_\psi C_{\text{sand}} \|} \right\rangle \frac{\nabla_\psi C_{\text{sand}}}{\| \nabla_\psi C_{\text{sand}} \|}, \tag{5.26}$$

with

$$\nabla_\psi C_{\text{sand}}(t) = 2\psi \int_\Omega (\psi - \psi_0) dx. \tag{5.27}$$

This new direction of descent, illustrated by Figure 5.11, describes a less optimal path to the minimum of $\mathcal{J}$, but ensures that $\nabla_\psi C_{\text{sand}}(t) = 0$, i.e. $C_{\text{sand}}(t) = 0$, for all $t \in [0, T]$. We can easily show that the new direction d and $\nabla_\psi C_{\text{sand}}$ are now orthogonal:

$$\langle d, \nabla_\psi C_{\text{sand}} \rangle = \left\langle \nabla_\psi \mathcal{J} - \left\langle \nabla_\psi \mathcal{J}, \frac{\nabla_\psi C_{\text{sand}}}{\| \nabla_\psi C_{\text{sand}} \|} \right\rangle \frac{\nabla_\psi C_{\text{sand}}}{\| \nabla_\psi C_{\text{sand}} \|}, \nabla_\psi C_{\text{sand}} \right\rangle = 0 \tag{5.28}$$

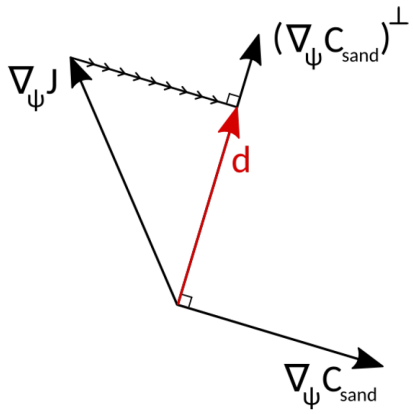


Figure 5.11 – Illustration of the new direction of descent in $\mathbb{R}^2$: the direction $\nabla_\psi \mathcal{J}$ is projected onto the orthogonal of C_{sand} to yield d.

3 Numerical Model

3.1 Presentation

In this section, we present the model OptiMorph, how to install and use it.

3.1.1 Workflow

Figure 5.12 illustrates the workflow of the OptiMorph model, with the associated hydrodynamic model. Prior to initiating the model, the user is required to establish

the initial configuration for the simulation. This includes the forcing data, the choice of hydrodynamic model, the seabed elevation data, and the constraints.

During each discrete time step, the forcing data is provided to the hydrodynamic model. This model then calculates the wave height over the cross-shore profile and thus provides the cost function $\mathcal{J}$ (or direction of descent d) used by OptiMorph's morphodynamic module. Using the imported sand characteristics, the new shape of the seabed is determined by minimizing the cost function $\mathcal{J}$ (or following the direction of descent d). Constraints are applied to the seabed either before or after the minimization takes place, and the new seabed is retained. At the next time step, the hydrodynamic model is fed a new forcing condition as well as the new seabed. This cycle continues over the course of the simulation, and illustrates the intricate interaction between the hydrodynamic and morphodynamic processes.

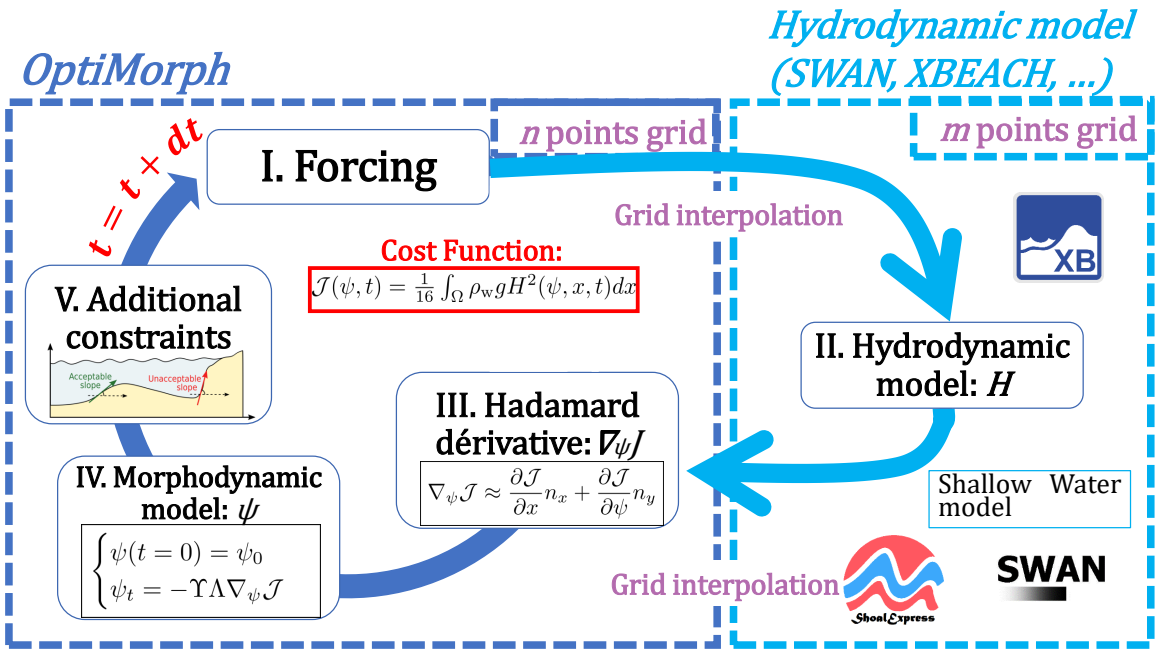


Figure 5.12 – OptiMorph workflow coupled with hydrodynamic model

3.1.2 Program Organization

The OptiMorph program is broken down into the following tree structure. There are 4 main folders:

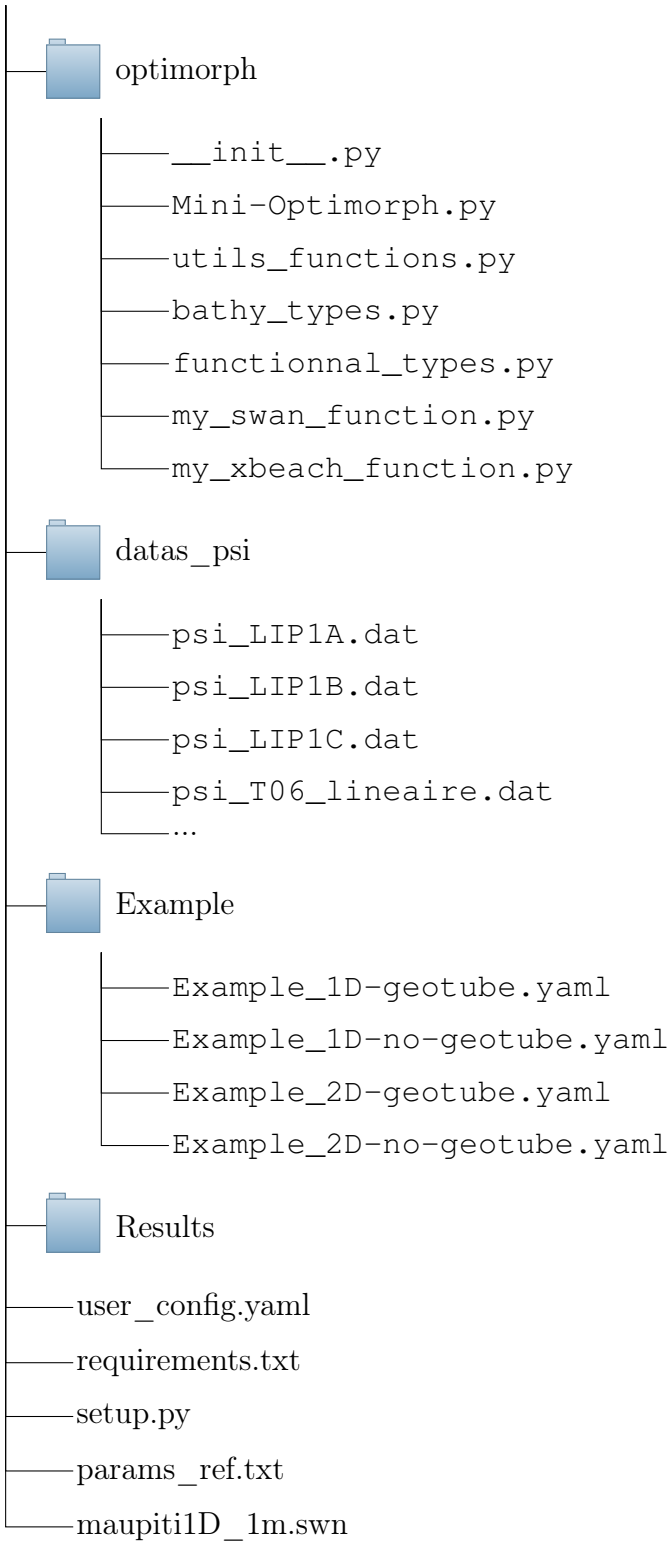
- a folder `optimorph` which is the heart of the program, all the code architecture (functions, governing equation, ...) is contained in it, the `Mini-Optimorph.py` program must be run to start the calculation;

- a `datas_psi` folder containing several types of bathymetry in `.dat` format;

- a `Results` folder where files and figures will be stored;

- a folder `Example` with files `user_config.yaml` already pre-configured.

To run a calculation, you need to change the parameters in the `user_config.yaml` file.


optimorph
__init__.py
Mini-Optimorph.py
utils_functions.py
bathy_types.py
functionnal_types.py
my_swan_function.py
my_xbeach_function.py
datas_psi
psi_LIP1A.dat
psi_LIP1B.dat
psi_LIP1C.dat
psi_T06_lineaire.dat
...
Example
Example_1D-geotube.yaml
Example_1D-no-geotube.yaml
Example_2D-geotube.yaml
Example_2D-no-geotube.yaml
Results
user_config.yaml
requirements.txt
setup.py
params_ref.txt
maupiti1D_1m.swn

3.2 Running OptiMorph

OptiMorph is installed in 3 stages. First, the basic OptiMorph code is installed. Next, we need numerical models such as SWAN or XBeach to couple our code to them. For these installations, we'll use Pagure, which makes them easier.

3.2.1 Installation of OptiMorph

The code we provide uses python version >3.7. To install it, please, download the last 1D version using the command line on terminal:

```
\$ git clone https://oauth2:github_pat_11A7K63PA0lZnQdd
↪  GwmZyy_KREFJ3NVsX2QZyR3oswTmNG99sFbEKMT2wUWYCVN5yLJKWOQKGQzqa1wtvn@github.com/r
↪  upont/OptiMorph-1-2D.git
```

If you don't have a terminal, you can always download the latest version from the following link: https://ronan-dupont.github.io/files/teaching/OptiMorph-1-2D.zip.

If you don't have `git` installed, you can install with the line:

```
\$ sudo apt install git
```

Then, you need to install the following modules on your computers (if you don't already have them):

- numpy
- matplotlib
- scipy
- pandas
- xarray
- pyyaml
- imageio
- Pillow

You can do it in one line with on unix / cmd (windows) terminal.

```
\$ pip install -r requirements.txt
```

With just this setup, we can run calculations using the Shoaling model. However, if we want to couple our OptiMorph model with SWAN, XBeach or other software, we need to install these programs.

If you have all the module installed and you'd like to make the first launch, you can run the following command:

```
\$ python3 optimorph/Mini-Optimorph.py
```

3.2.2 Installation of PAGURE (to install SWAN and XBEACH)

The first step is to install a software program called PAGURE [1], developed by a former GLADYS doctoral student and post-doctoral fellow (Fabien Rétif [2]). This software will collect all the libraries needed to run the digital tools, and link them in their correct version with the compiler chosen to compile on the cluster.

First, we connect to the cluster:

```
\$ ssh e_gcl-XX@muse-login01.hpc-lr.univ-montp2.fr
```

From now on, all the commands presented in this section are to be entered on the cluster, not on your local machine. We use `git` – which we'll see in detail later – to retrieve pagure.

1. https://github.com/fretif/pagure
2. https://www.fabienretif.com

```
\$ cd
\$ mkdir install-softs
\$ cd install-softs
\$ mkdir pagure.git
\$ cd pagure.git
\$ git clone https://github.com/fretif/pagure.git .
Cloning into '.'...
remote: Enumerating objects: 2000, done.
remote: Counting objects: 100% (334/334), done.
remote: Compressing objects: 100% (231/231), done.
remote: Total 2000 (delta 229), reused 200 (delta 103), pack-reused
↪  1666
Receiving objects: 100% (2000/2000), 4.67 MiB | 0 bytes/s, done.
Resolving deltas: 100% (1475/1475), done.
\$ ./pagure.sh   # affiche un message de PAGURE par défaut
```

PAGURE is now installed in the directory `/install-softs/pagure.git`.

3.2.2.1 Configuring your work environment

Before using PAGURE, it's important to configure your working environment. To do this, we're going to use an environment manager called `module` (yes, that's a funny name!). To illustrate the usefulness of an environment manager, I suggest the following scenario:

Imagine you've installed version 2.18 of QGis on your computer and your colleague shares a file with you that works with version 3.20. What do you do? You can update QGis to version 3.20. But you run the risk that your files won't open with the new version, or that some features have disappeared. This is where the environment manager comes into play, allowing you to have both versions co-exist on your system at the same time. You can then choose to load version 2.18 into your working environment and switch to another version at any time.

An environment manager is useful when you need complete control over your working environment and the flexibility to run multiple versions of the same software. This is perfectly justified in the case of a cluster or a workstation dedicated to numerical computation.

In most computers, such as MESO, the `module` tool is supplied directly by the cluster administrators. On your local workstation, the `module` tool will probably not be installed, but PAGURE will detect it and install it automatically.

Without further ado, let's start using the `module` tool to configure our working en-

vironment on MESO.

```
\$ module avail

------------------- /usr/share/Modules/modulefiles
 ↪  ---------------------------------------
dot         module-git  module-info modules     null        use.own

------------------ /trinity/shared/modulefiles/modulegroups
 ↪  --------------------------------
cv-admin    cv-advanced cv-local    cv-standard local

------------------ /trinity/shared/modulefiles/local
 ↪  --------------------------------------
.....
----------------------------------
 ↪  /trinity/shared/modulefiles/cv-standard -----------------
....
intel/itac/64/2020.4.912            intel/mkl/32/2017.1.132
intel/compiler/32/2016.3.210        intel/compiler/64/2016.3.210
intel/compiler/32/2017.1.132        intel/compiler/64/2017.1.132
intel/compiler/32/2020.4.912        intel/compiler/64/2020.4.912
gcc/4.9.3(default)                  intel/mkl/64/2020.4.912
gcc/6.1.0                           intel/omnipath/64/libpsm2-10.3.8-3
gcc/7.5.0                           git/2.9.3
gcc/8.5.0                           intel/mpi/64/2017.1.132
gdb/7.11                            intel/mpi/64/2020.4.912
....
```

This command lists all the libraries/software available on the cluster, with their versions and sometimes the name of the compiler editor used (GCC or Intel).

Let's start by resetting the working environment with the command :

```
\$ module purge
```

Then we'll load the modules `cv-standard` `use.own` and `intel/compiler/64/2017.1.132` with the command line:

```
\$ module load cv-standard use.own intel/compiler/64/2017.1.132
```

Let's check that the libraries have been loaded in our working environment:

```
\$ module list
Currently Loaded Modulefiles:
1) cv-standard      2) use.own      3) intel/compiler/64/2017.1.132
\$  ifort --version
ifort (IFORT) 17.0.1 20161005
Copyright (C) 1985-2016 Intel Corporation.  All rights reserved.
```

We've loaded the 2017 version of the `Intel` brand compiler. Our environment is ready!

3.2.2.2 Installing the SWAN model

To install the `SWAN` template, you must first configure your working environment with the 2017 version of the `INTEL` compiler (see section 3.2.2.1). Next, we'll run `PAGURE` with a set of arguments corresponding to this model.

```
\$ ./pagure.sh --prefix=/home/e_gcl-XX/softs --system=cluster
--compiler=intel --filter=SWAN
 ↪  --module-dir=/home/e_gcl-XX/privatemodules
```

This command will compile and install a set of modules (in the sense of `module` software) in the `privatemodules` directory of your personal environment, download the `SWAN` software and compile and link it to these libraries, then install it in the `soft` directory of your personal environment.

On startup, `PAGURE` summarizes information about the working environment it has detected and the software it will install:

```
[  INFO  ]  system is set to cluster
[  INFO  ]  prefix is set to /home/e_gcl-XX/softs
[  INFO  ]  module dir is set to /home/e_gcl-XX/privatemodules
[  INFO  ]  Installation mode is set to auto
[  INFO  ]  Force to download is set to 0
[  INFO  ]  Force to reinstall is set to 0
[  INFO  ]  Auto-remove is set to 1
[  INFO  ]  Automatic installation of mandatory libraries is set to 1
[  INFO  ]  When using a filter, show old version is set to 1
[  INFO  ]  Python interpreter is set to python2.7
.....................
[  INFO  ]  compiler is set to INTEL 17
[  INFO  ]  MPI library is set to mpich321
.....................
[  OK  ]  Make dir prefix
.....................
[  INFO  ]  The following libraries are pre-selected to be installed :
[  INFO  ]  mpich 3.2.1
[  INFO  ]  zlib 1.2.11 (needed by HDF5)
[  INFO  ]  parallel-netcdf 1.12.1 (needed by Netcdf 4.8.0)
[  INFO  ]  hdf5 1.10.5 (with parallel I/O)
[  INFO  ]  netcdf 4.8.0 (version C - need HDF 1.10.5 and
↪  Parallel-Netcdf 1.12.1)
[  INFO  ]  netcdf 4.5.3 (version Fortran - need Netcdf-C 4.8.0, HDF
↪  1.10.5 and Parallel-Netcdf 1.12.1)
[  INFO  ]  swan 41.31
.....................
[  OK  ]  We are now ready to install. Please check the information
↪  above
.....................
Everything is OK ? Press Enter to continue or press q to quit
```

At this point, PAGURE asks whether the information detected is correct before continuing its execution. You should therefore check that :

- the system detected is indeed that of a cluster

```
[  INFO  ]  system is set to cluster
```

- the prefix path, i.e. where all libraries useful to SWAN will be installed, is `/home/e_gcl-XX/softs`.

```
[  INFO  ]  prefix is set to /home/e_gcl-XX/softs
```

- the path of the modules directory, i.e. where the modules will be installed to load the SWAN libraries in your work environment, is /home/e_gcl-XX/privatemodules.

```
[  INFO  ]  module dir is set to /home/e_gcl-XX/privatemodules
```

- the detected compiler is the 2017 version of Intel.

```
[  INFO  ]  compiler is set to INTEL 17
```

- the parallel computing library (MPI) is Mpich

```
[  INFO  ]  MPI library is set to mpich321
```

If everything is OK, you can press Enter on your keyboard and wait for PAGURE to finish installing SWAN.

```
# un temps looooooooooooooooooooooonnnnnnnnnng! (sans doute 1h la
 ↪  première fois), puis:
[  INFO  ]  Removing archive file and source files
[  OK  ]  Install swan 41.31
[  OK  ]  Congratulation, you did it
```

If all has gone well, you can check the installation by listing the libraries with the module tool.

```
\$ module avail
....
-----------------------------------/home/e_gcl-XX/privatemodules
 ↪  ---------------------------
netcdf-fortran/hdf5.110/mpich321/icc17/4.5.3
 ↪  hdf5/mpich321/icc17/1.10.5
netcdf-c/hdf5.110/mpich321/icc17/4.8.0
 ↪  parallel-netcdf/mpich321/icc17/1.12.1
zlib/icc17/1.2.11                                mpich/icc17/3.2.1
swan/mpich321/icc17/41.31
```

So you have a new module swan/mpich321/icc17/41.31

3.2.2.3 Installing the XBEACH model

To install the XBEACH template, we need to start by loading a new working environment.

Let's start by resetting the working environment with the command :

```
\$ module purge
```

Then we'll load the modules `cv-standard use.own` and `gcc/7.5.0` with the following command:

```
\$ module load cv-standard use.own gcc/7.5.0
```

Let's check that the libraries have been loaded in our working environment:

```
\$ module list
Currently Loaded Modulefiles:
1) cv-standard      2) use.own      3) gcc/7.5.0
\$  gcc --version
gcc (GCC) 7.5.0
Copyright © 2017 Free Software Foundation, Inc.
Ce logiciel est un logiciel libre; voir les sources pour les
 ↪  conditions de copie.
Il n'y a AUCUNE GARANTIE, pas même pour la COMMERCIALISATION ni
 ↪  L'ADÉQUATION À UNE TÂCHE PARTICULIÈRE.
```

We've loaded the GNU/GCC brand compiler in version 7.5. Our environment is ready!

To install the XBEACH model in its sequential version, we'll run PAGURE with a set of arguments corresponding to this model.

```
\$ ./pagure.sh --prefix=/home/e_gcl-XX/softs --system=cluster
--compiler=gnu --filter=XBEACH
 ↪  --module-dir=/home/e_gcl-XX/privatemodules
```

This command will compile and install a set of modules (in the sense of `module` software) in the `privatemodules` directory of your personal environment, download the XBEACH software and compile and link it to these libraries, then install it in the `soft` directory of your personal environment.

On startup, PAGURE summarizes the information about the working environment it has detected and the software it will install:

```
[  INFO  ]  system is set to cluster
[  INFO  ]  prefix is set to /home/e_gcl-XX/softs
[  INFO  ]  module dir is set to /home/e_gcl-XX/privatemodules
[  INFO  ]  Installation mode is set to auto
[  INFO  ]  Force to download is set to 0
[  INFO  ]  Force to reinstall is set to 0
[  INFO  ]  Auto-remove is set to 1
[  INFO  ]  Automatic installation of mandatory libraries is set to 1
[  INFO  ]  When using a filter, show old version is set to 1
[  INFO  ]  Python interpreter 3.7 will be installed
.....................
[  INFO  ]  compiler is set to GNU 7.5
[  WARNING  ]  No MPI library
.....................
[  OK  ]  Make dir prefix
.....................
[  INFO  ]  The following libraries are pre-selected to be installed :
[  INFO  ]  python 3.7
[  INFO  ]  setuptools 57.0.0 (Python module)
[  INFO  ]  mako 1.2.0 (Python module)
[  INFO  ]  zlib 1.2.11 (needed by HDF5)
[  INFO  ]  hdf5 1.10.5
[  INFO  ]  netcdf 4.8.0 (version C - need HDF 1.10.5)
[  INFO  ]  netcdf 4.5.3 (version Fortran - need Netcdf-C 4.8.0 and
 ↪  HDF 1.10.5)
[  INFO  ]  xbeach rev5920 (sequential version)
.....................
[  OK  ]  We are now ready to install. Please check the information
 ↪  above
.....................
Everything is OK ? Press Enter to continue or press q to quit
```

At this point, PAGURE asks whether the information detected is correct before continuing its execution. You should therefore check that :

- the system detected is indeed that of a cluster

  ```
  [  INFO  ]  system is set to cluster
  ```

- the prefix path, i.e. where all libraries useful to XBEACH will be installed, is `/home/e_gcl-XX/softs`

```
[  INFO  ]  prefix is set to /home/e_gcl-XX/softs
```

- the path of the modules directory, i.e. where the modules will be installed to load the XBEACH libraries in your work environment, is /home/e_gcl-XX/privatemodules.

```
[  INFO  ]  module dir is set to /home/e_gcl-XX/privatemodules
```

- the detected compiler is the GNU/GCC compiler in version 7.5

```
[  INFO  ]  compiler is set to GNU 7.5
```

If everything is OK, you can press Enter on your keyboard and wait for PAGURE to finish installing XBEACH. Compilation may take some time.

```
# un temps looooooooooooooooooooooonnnnnnnnnng! (sans doute 1h la
↪  première fois), puis:
.....................
Type the absolute path of the archive file 'xbeach-rev5920.zip' :
```

At this stage, you need to specify the path of the 'xbeach-rev5920.zip' file distributed to you (which can also be found at https://drive.google.com/file/d/19Ngh9vfnkCzLdVgXjcCL view?usp=sharing). For example, if you copied it to your home directory, you would specify: /home/e_gcl-XX

```
# un temps looooooooooooooooooooooonnnnnnnnnng! (sans doute 1h la
↪  première fois), puis:
[  INFO  ]  Removing archive file and source files
[  OK  ]  Install xbeach rev5920 (sequential version)
[  OK  ]  Congratulation, you did it
```

If all has gone well, you can check the installation by listing the libraries with the module tool.

```
\$ module avail
....
-----------------------------------/home/e_gcl-XX/privatemodules
 ↪  --------------------------
netcdf-fortran/hdf5.110/mpich321/icc17/4.5.3
 ↪  hdf5/mpich321/icc17/1.10.5
netcdf-c/hdf5.110/mpich321/icc17/4.8.0
 ↪  parallel-netcdf/mpich321/icc17/1.12.1
zlib/icc17/1.2.11                                mpich/icc17/3.2.1
swan/mpich321/icc17/41.31                        xbeach/gcc75/rev5920
```

So you have a new module `xbeach/gcc75/rev5920` corresponding to the sequential version of XBEACH.

3.2.3 Input File

The input file for the OptiMorph code is the `user_config.yaml`. It is presented as follows:

user_config.yaml

```yaml
dirname: Example_1D-convexe_tide # dirname to save figs
 ↪  and data
figname: Example Simulation 1D of storm in convexe
 ↪  bathymetry with tide # simulation name appear on figs
debug: False # this mode plot some interesting values
makeGifs: True # make gifs
T0: 6 # wave period
H0: 1.5 # offshore wave height
h0: 10
nwater: 600 # in the water
nsand: 140 # in the sand
n_iteration: 1000
ifre: 50 # save and plot every ifre iteration
mobility: 0.004
bathy_type: 1 # [0:18]
slope_max: 0.2 #
id_cost_fct: 1 #  cost function [1:12]
hydro_mode: 1 #  : 0 = shoaling, 1 = swan, 2 = XBeach
dynamic: False
gamma: 0.55
coef_maree: 60 # tide coefficient
u_maree: 6.1  # valeur moyenne du marnage
maree_duration: 12.5 # periode de maree
geotube:
  state: False
  position_x: 110 # geotube position x [m]
  position_y: 10 # geotube position y [m] ONLY FOR 2D MODE
  length: 6    # geotube length [m]
  height: 2   # geotube height [m]
two_dimension:
  state: False
  n_i: 300
  n_j: 60
  L_x: 600
  L_y: 20
```

with the description in the table 5.1 below.

Parameter	Type	Name	Description	Unit
dirname	String	-	Directory name	-
figname	String	-	Figure name	-
debug	Boolean	Debug mode	This mode plot some interesting values	-
makeGifs	Boolean	Make Gifs	This make a gif	-
T0	Float	T_0	Wave period	m
H0	Float	H_0	Offshore wave height	m
h0	Float	h_0	Depth of closure	m
Omega	Integer	Ω	Domain size	m
n_iteration	Integer	$n_{iteration}$	Number of iterations	-
ifre	Integer	-	Save/Plot every ifre	-
mobility	Float	Υ	Mobility parameter	$m.s.kg^{-1}$
bathy_type	Integer	-	Sea Bottom type ranging [0,18]	-
slope_max	Float	M_{slope}	Maximum slope	-
id_cost_fct	Integer	-	Cost Function type ranging [1,12]	-
hydro_mode	Integer	-	Hydrodynamic mode, 0 = shoaling, 1 = swan, 2 = XBeach	-
dynamic	Boolean	-	Static (0) or Dynamic (1) forcing	-
gamma	Float	γ	Breaking criterion	-
coef_maree	Integer	C_{tide}	Tidal coefficient	-
u_maree	Float	M_{ref}	The reference tidal range	m
maree_duration	Float	T_{tide}	Tide duration	h
geotube:				
state	Boolean	-	Activate (True) or not (False)	-
position_x	Integer	-	Position x of geotube	m
position_y	Integer	-	Position y of geotube	m
length	Integer	-	Geotube length	m
height	Integer	-	Geotube height	m
two_dimension:				
state	Boolean	-	Activate (True) or not (False)	-
n_i	Integer	-	Number of points on x axis	-
n_j	Integer	-	Number of points on y axis	-
L_x	Integer	L_x	Length of x axis	m
L_y	Integer	L_y	Length of y axis	m

Table 5.1 – Input parameters

This file was created as described below. However, you can easily add parameters by opening the file `optimorph/Mini-Optimorph.py`.

4 Applications

In this section, the seabed is described as a simple linear function over the cross-shore profile. First, we simulate the results over a homogeneous sandy seabed, then we look at introducing submerged structures designed to limit wave activity at the coast. Finally, we study this last case in 2D on a linear seabed also inclined along y.

4.1 1D Linear Seabed Beach Configuration using Hadamard approach with SWAN

The applications focus on 3 different cases. These 3 cases aim to show how to use the code with the 3 different hydrodynamics as well as on 3 different configurations.

4.1.1 Setting

The initial cross-shore configuration is given in Figure 5.13: the domain measures 740 m, the mean water level is set at 10 m and we apply a storm profile to the seabed, given by the top left graph of Figure 5.13 . Here we consider a homogeneous sandy seabed, and therefore the mobility of the seabed Υ and the maximal slope parameter M_{slope} slope are assumed constant over the cross-shore profile Ω.

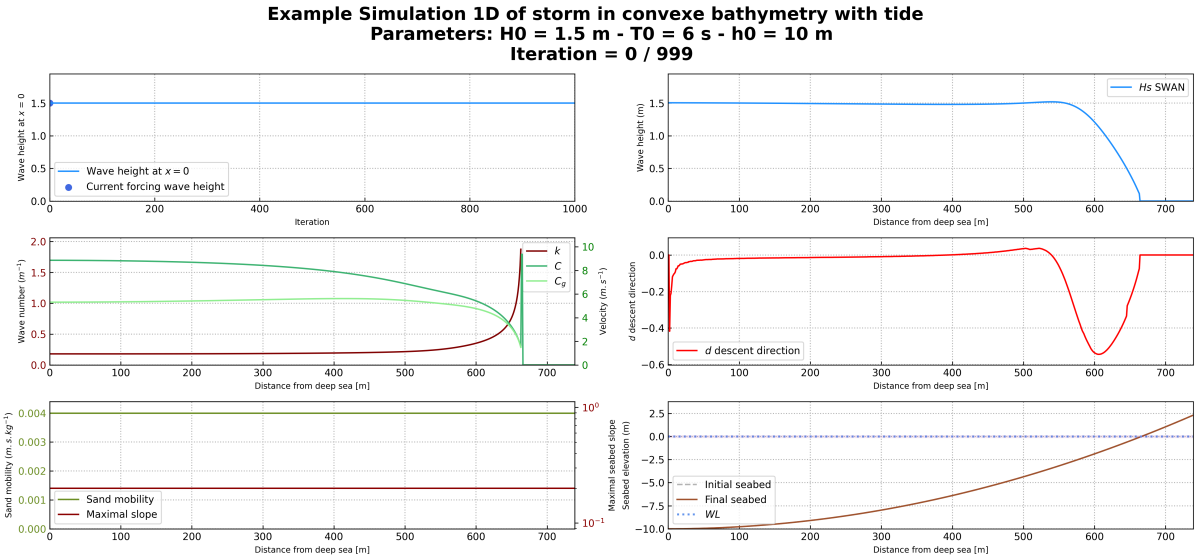


Figure 5.13 – Initial sandy beach configuration

4.1.2 Input files

The input file is present in the `user_config.yaml` file and is configured as follows. It can also be found at the location `Example/Example_1D-no-geotube_tide.yaml`.

user_config.yaml

```yaml
dirname: Example_1D-convexe_tide # dirname to save figs
 ↪  and data
figname: Example Simulation 1D of storm in convexe
 ↪  bathymetry with tide # simulation name appear on figs
debug: False # this mode plot some interesting values
makeGifs: True # make gifs
T0: 6 # wave period
H0: 1.5 # offshore wave height
h0: 10
nwater: 600 # in the water
nsand: 140 # in the sand
n_iteration: 1000
ifre: 50 # save and plot every ifre iteration
mobility: 0.004
bathy_type: 1 # [0:18]
slope_max: 0.2 #
id_cost_fct: 1 #  cost function [1:12]
hydro_mode: 1 #  : 0 = shoaling, 1 = swan, 2 = XBeach
dynamic: False
gamma: 0.55
coef_maree: 60 # tide coefficient
u_maree: 6.1  # valeur moyenne du marnage
maree_duration: 12.5 # periode de maree
geotube:
  state: False
  position_x: 110 # geotube position x [m]
  position_y: 10 # geotube position y [m] ONLY FOR 2D MODE
  length: 6    # geotube length [m]
  height: 2   # geotube height [m]
two_dimension:
  state: False
  n_i: 300
  n_j: 60
  L_x: 600
  L_y: 20
```

4.1.3 Load SWAN and Run OptiMorph on Cluster

To launch OptiMorph with SWAN, you need to load SWAN into your modules. Here are a few commands that will enable you to run OptiMorph on the cluster without any problems:

```
\$ module purge
\$ module load use.own swan/mpich321/icc17/41.31
\$ module load python/3.7.2
\$ pip install --upgrade pip
\$ pip install -e .
\$ pip install -U matplotlib
\$ python3 optimorph/Mini-Optimorph.py
```

You can also create a bash to launch the file on slurm with the file below:

run.cmd

```
#!/bin/bash

# Example of running python script with a job array

#SBATCH -J Run_test
#SBATCH -p gm_gladys
#SBATCH --account=shoremotion
#SBATCH -c 1                              # one CPU core per task
#SBATCH -o console.out
#SBATCH -e erreur.out
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --ntasks-per-node 1
#SBATCH --ntasks-per-core 1

# Run python script with a command line argument
srun python optimorph/Mini-Optimorph.py
```

then run it via the command:

```
\$ sbash run.cmd
```

4.1.4 Results

At the end of the simulation, we get the following results of Figure 5.13, 5.14, 5.15, 5.16, 5.17, 5.18, 5.19 and 5.20.

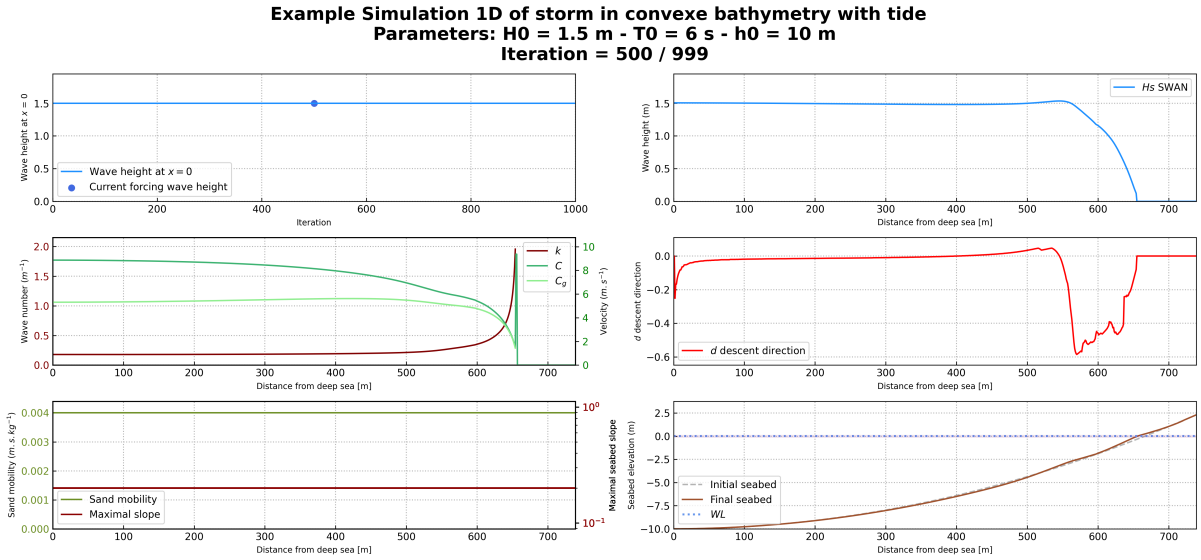


Figure 5.14 – Results halfway through the simulation

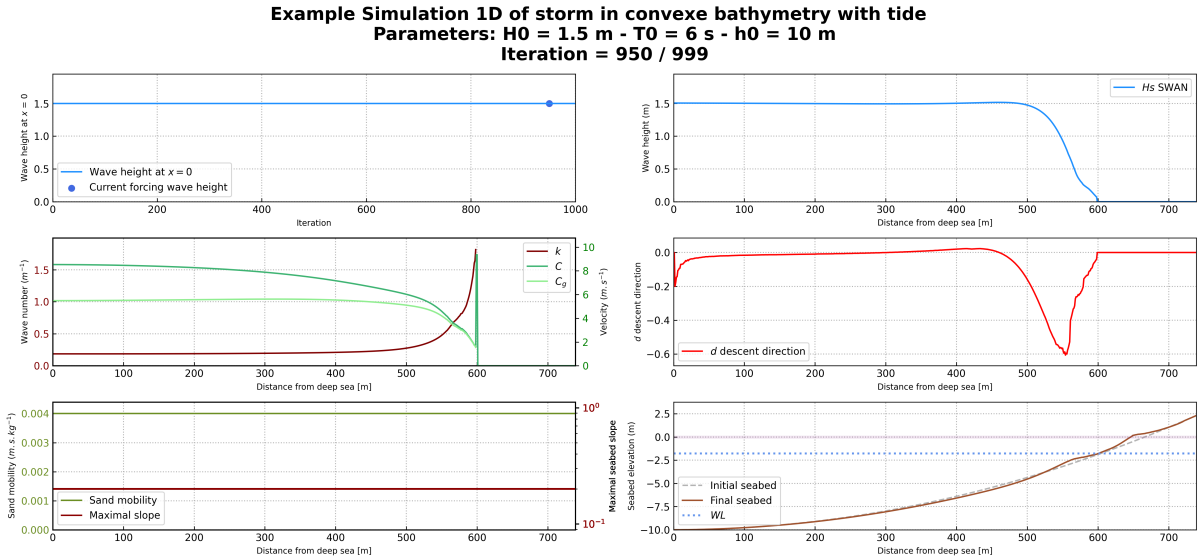


Figure 5.15 – Results at the end of the simulation

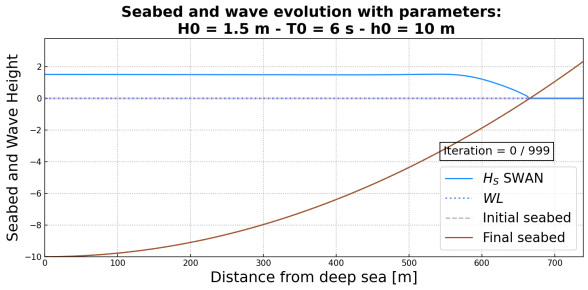


Figure 5.16 – Initial seabed at the beginning of the simulation

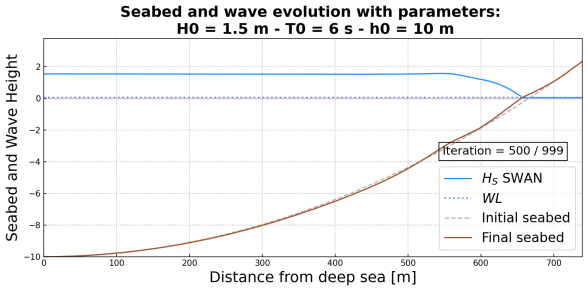


Figure 5.17 – Seabed halfway through the simulation

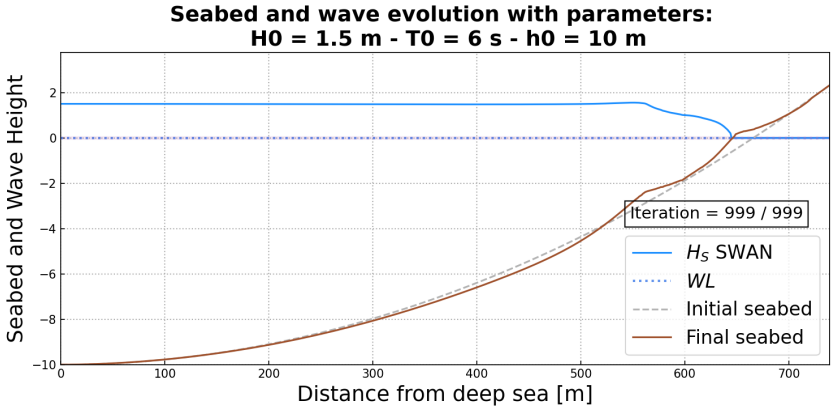


Figure 5.18 – Final seabed at the end of the simulation

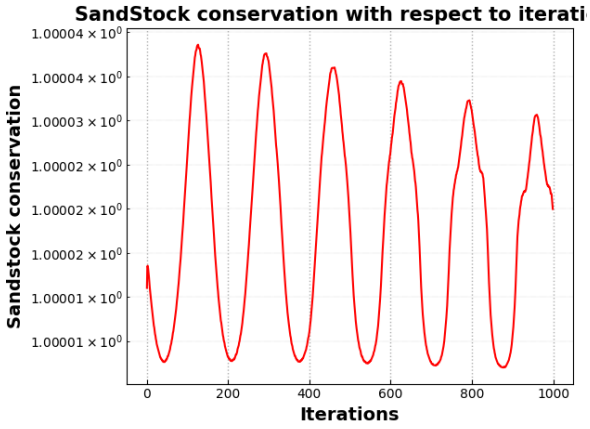


Figure 5.19 – Variation of the sandstock over time

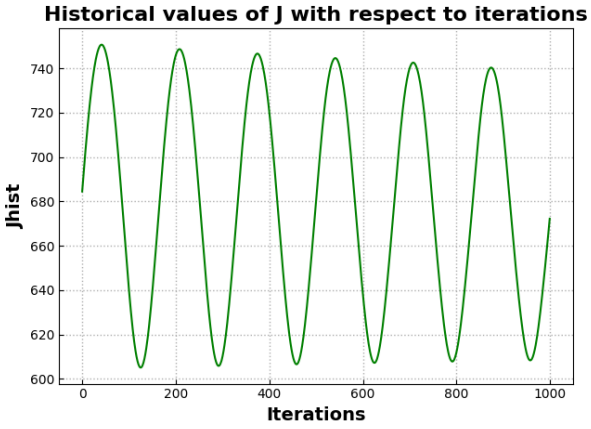


Figure 5.20 – Variation of d over time

A thorough analysis of the results of OptiMorph can be found in (Ronan Dupont et al. 2023) for an experimental flume configuration and (Ronan Dupont et al. 2022) for a linear seabed. Sand conservation has an error of the order of 10^{-5}.

4.2 1D Linear Seabed Beach with GeoTube using Hadamard approach with XBeach

4.2.1 Setting

In this simulation, we introduce a submerged solid structure. To do this, we modify the seabed profile, as well as the sand mobility parameter Y and the maximal slope parameter M_{slope}, which are no longer constant over the cross-shore profile. In the case of the mobility parameter, no movement can occur at the location of the structures, i.e. Y = 0 where the breakwater is positioned. Similarly, the maximal slope parameter has also been modified to locally deactivate the slope constraint over the structure. Figure 5.21 shows the new initial configuration incorporating a submerged breakwater located at x = 600 m.

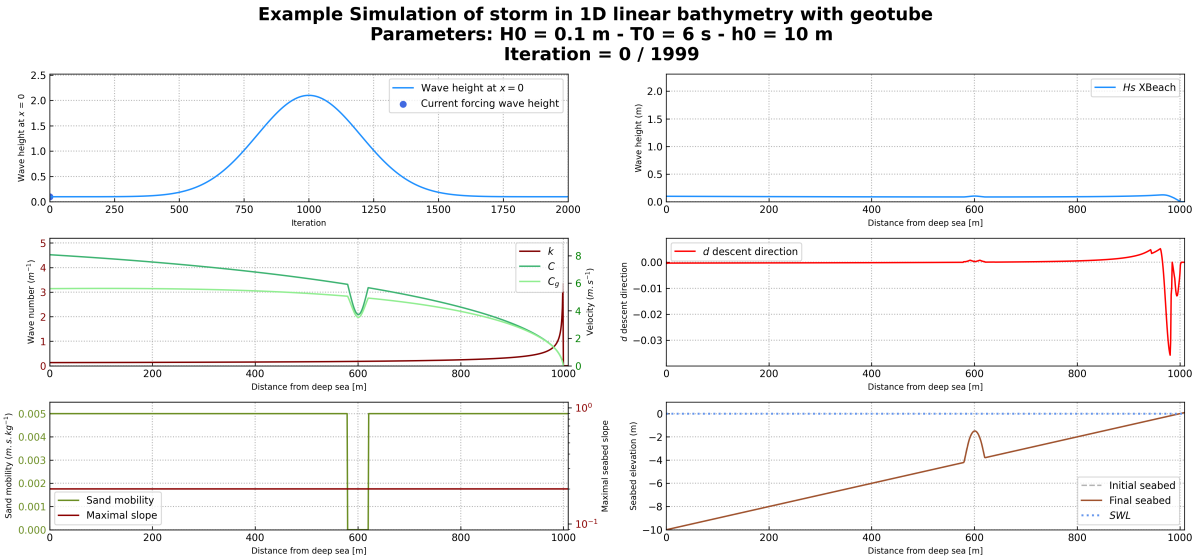


Figure 5.21 – Initial sandy beach configuration with a submerged breakwater located at x = 600 m

4.2.2 Input Files

The input file is present in the `user_config.yaml` file and is configured as follows. It can also be found at the location `Example/Example_1D-geotube.yaml`.

user_config.yaml

```yaml
dirname: 1D-geotube_example # dirname to save figs
figname: Example Simulation of storm in 1D linear
 ↪  bathymetry with geotube # simulation name
debug: False # this mode plot some interesting values
makeGifs: True # make gifs
T0: 6 # wave period
H0: 2 # offshore wave height
h0: 10
Omega: 1000
n_iteration: 2000
ifre: 50 # save and plot every ifre iteration
mobility: 0.005
bathy_type: 0 # [0:18]
slope_max: 0.2 #
id_cost_fct: 1 #  cost function [1:12]
hydro_mode: 2 #  : 0 = shoaling, 1 = swan, 2 = XBeach
dynamic: True
gamma: 0.55
coef_maree: 0 # tide coefficient
u_maree: 6.1  # valeur moyenne du marnage
maree_duration: 12.5 # periode de maree
geotube:
  state: True
  position_x: 600 # geotube position x [m]
  position_y: 10 # geotube position y [m] ONLY FOR 2D MODE
  length: 40    # geotube length [m]
  height: 2.5   # geotube height [m]
two_dimension:
  state: False
  n_i: 300
  n_j: 60
  L_x: 600
  L_y: 20
```

4.2.3 Load XBeach and Run OptiMorph on Cluster

To launch OptiMorph with XBeach, you need to load XBeach into your modules. Here are a few commands that will enable you to run OptiMorph on the cluster without any problems:

```
\$ module purge
\$ module load use.own xbeach/gcc75/rev5920
\$ module load python/3.7.2
\$ pip install --upgrade pip
\$ pip install -e .
\$ pip install -U matplotlib
\$ python3 optimorph/Mini-Optimorph.py
```

You can also create a bash to launch the file on slurm with the file below:

run.cmd

```
#!/bin/bash

# Example of running python script with a job array

#SBATCH -J Run_test
#SBATCH -p gm_gladys
#SBATCH --account=shoremotion
#SBATCH -c 1                              # one CPU core per task
#SBATCH -o console.out
#SBATCH -e erreur.out
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --ntasks-per-node 1
#SBATCH --ntasks-per-core 1

# Run python script with a command line argument
srun python optimorph/Mini-Optimorph.py
```

then run it via the command:

```
\$ sbash run.cmd
```

4.2.4 Results

At the end of the simulation, we get the following results of Figure 5.21, 5.22, 5.23, 5.24, 5.25, 5.26, 5.27 and 5.28.

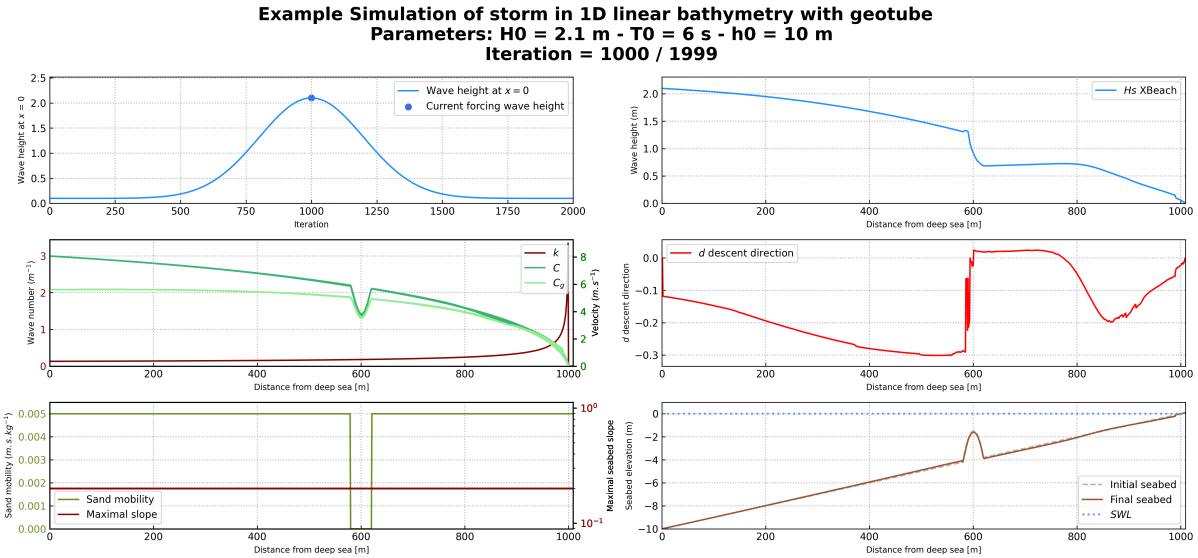


Figure 5.22 – Results halfway through the simulation

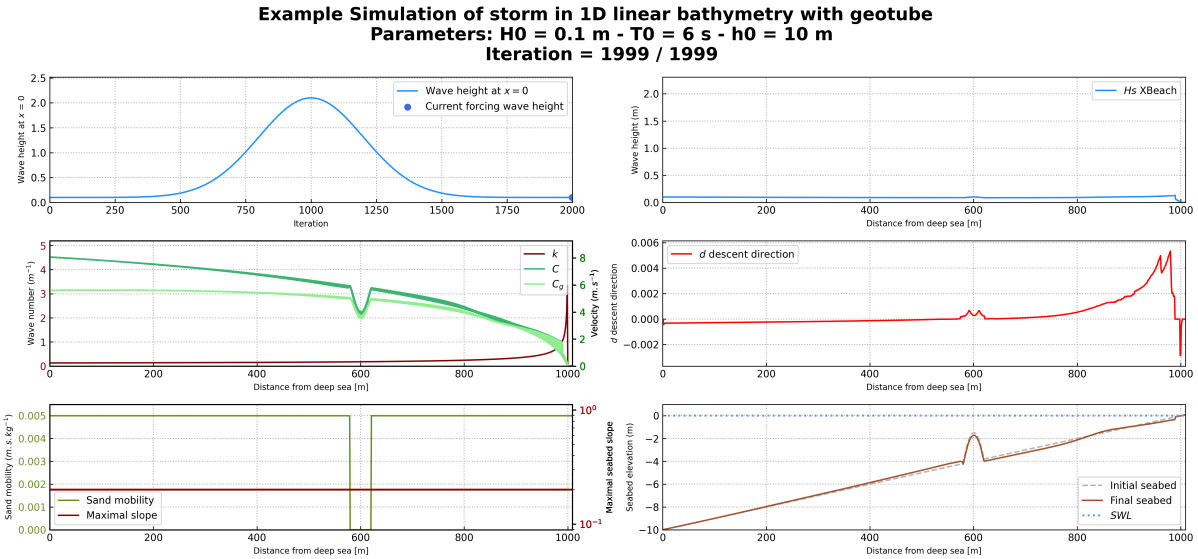


Figure 5.23 – Results at the end of the simulation

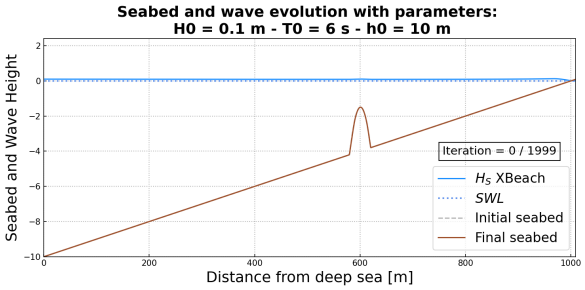


Figure 5.24 – Initial seabed at the beginning of the simulation

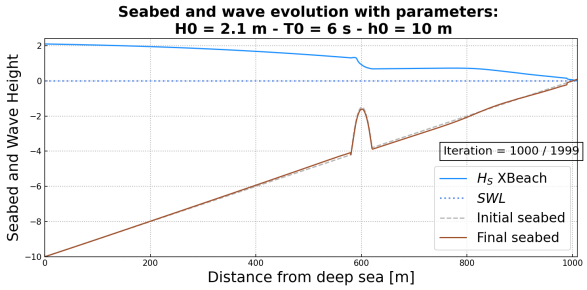


Figure 5.25 – Seabed halfway through the simulation

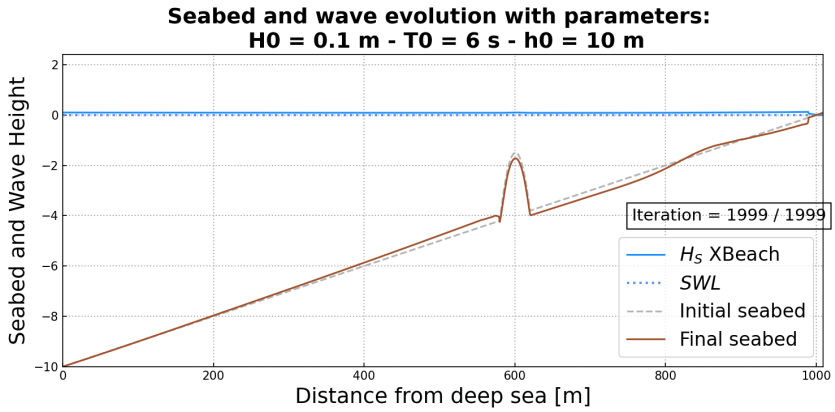


Figure 5.26 – Final seabed at the end of the simulation

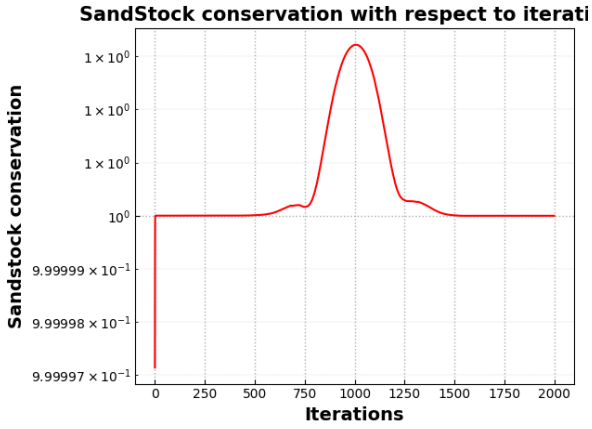


Figure 5.27 – Variation of the sandstock over time

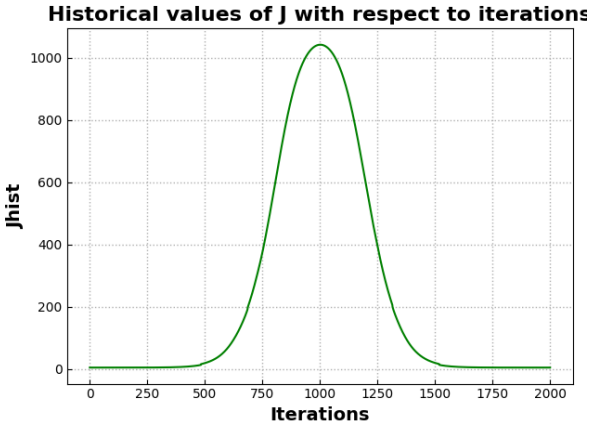


Figure 5.28 – Variation of d over time

We observe that in the case where there is a geotube (figure 5.26), there is less morphdynamic displacement than without a geotube (figure 5.26). Sand conservation has an error of the order of 10^{-5}.

4.3 2D Linear Seabed Beach Configuration using Hadamard approach with Shoaling

4.3.1 Setting

In this simulation, we continue with a solid submerged structure but in 2D. To do this, we modify the seabed profile, as well as the sand mobility parameter Y and the maximal slope parameter M_{slope}, which are no longer constant over the cross-shore profile. In the case of the mobility parameter, no movement can occur at the location of the structures, i.e. $Y = 0$ where the breakwater is positioned. Similarly, the maximal slope parameter has also been modified to locally deactivate the slope constraint over the structure. Figure 5.29 shows the new initial configuration incorporating a submerged breakwater located at $x = 600$ m.

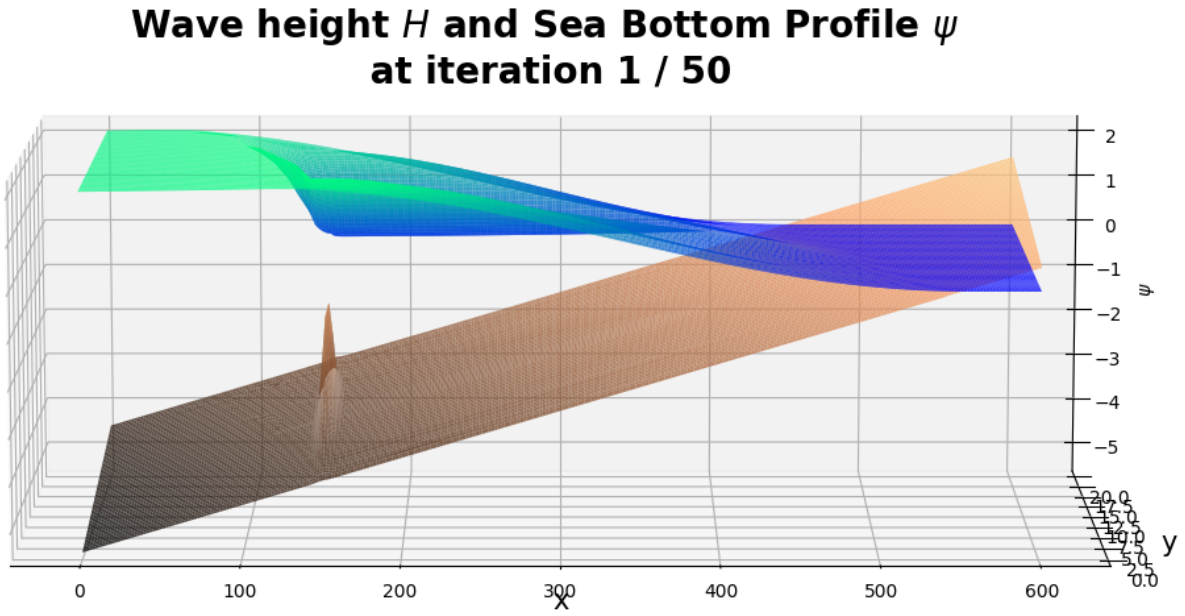


Figure 5.29 – Initial sandy beach configuration with a submerged breakwater located at $x = 150$ m

4.3.2 Input Files

The input file is present in the `user_config.yaml` file and is configured as follows. It can also be found at the location `Example/Example_2D-geotube.yaml`.

user_config.yaml

```yaml
dirname: plot_test_2D # dirname to save figs
figname: Simulation of storm in linear bathymetry with
 ↪  geotube # simulation name
debug: False # this mode plot some interesting values
makeGifs: True # make gifs
T0: 6 # wave period
H0: 2 # offshore wave height
h0: 5.5
Omega: 600
n_iteration: 50
ifre: 5 # save and plot every ifre iteration
mobility: 0.05
bathy_type: 0 # [0:18]
slope_max: 0.2 #
id_cost_fct: 1 #  cost function [1:12]
hydro_mode: 0 #  : 0 = shoaling, 1 = swan, 2 = XBeach
dynamic: True
gamma: 0.55
coef_maree: 0 # tide coefficient
u_maree: 6.1  # valeur moyenne du marnage
maree_duration: 12.5 # periode de mareex
geotube:
  state: True
  position_x: 150 # geotube position x [m]
  position_y: 10 # geotube position y [m] ONLY FOR 2D MODE
  length: 40    # geotube length [m]
  height: 2.5   # geotube height [m]
two_dimension:
  state: True
  n_i: 300
  n_j: 60
  L_x: 600
  L_y: 20
```

4.3.3 Results

At the end of the simulation, we get the following results of Figure 5.31, 5.32, 5.29, 5.33 and 5.34.

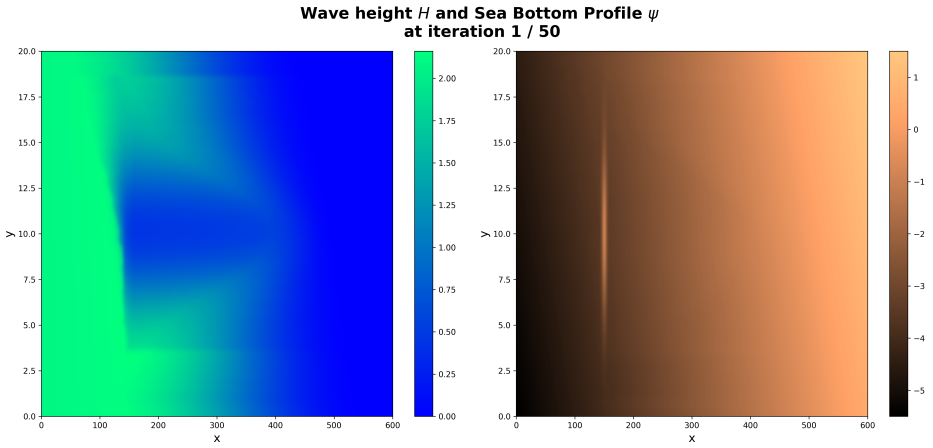


Figure 5.30 – Initial sandy beach configuration with a submerged breakwater located at $x = 150$ m

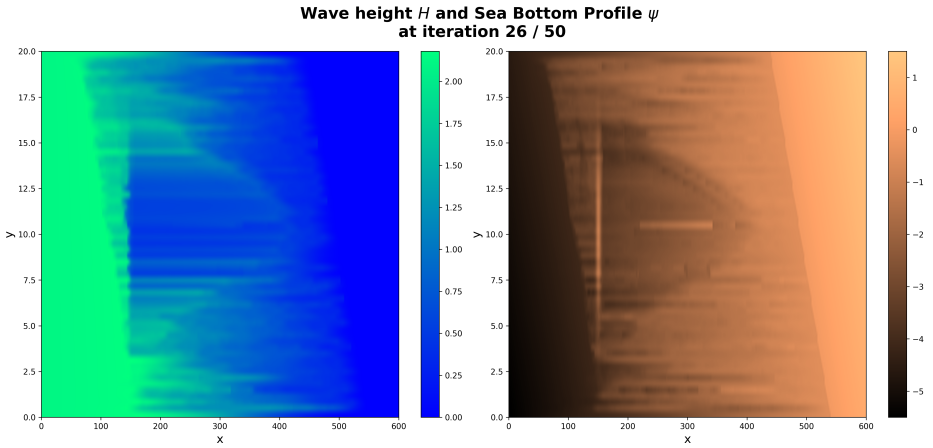


Figure 5.31 – Seabed halfway through the simulation

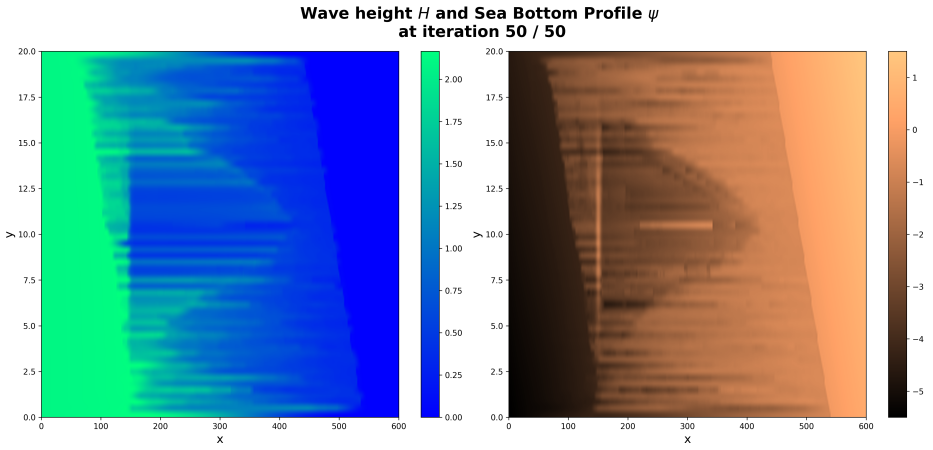


Figure 5.32 – Final seabed at the end of the simulation

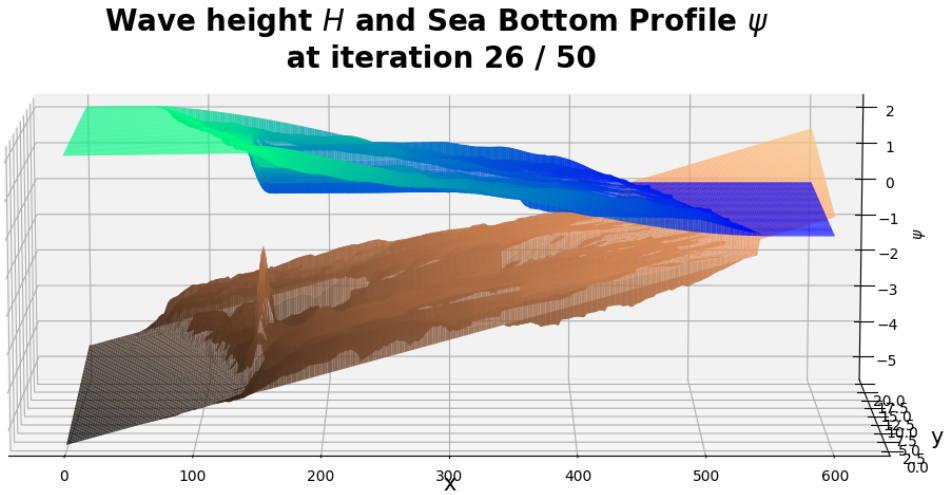


Figure 5.33 – Seabed halfway through the simulation

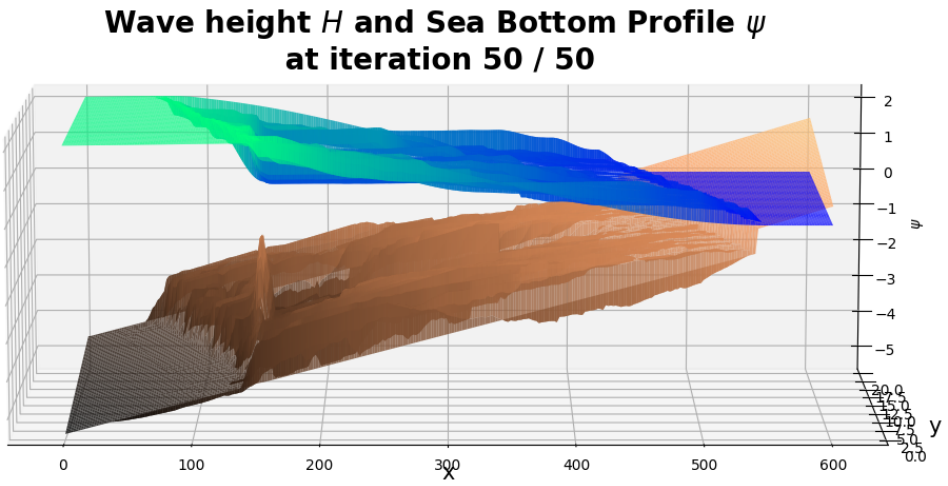


Figure 5.34 – Final seabed at the end of the simulation

A thorough analysis of the results of OptiMorph can be found in chapter 4 of the thesis for the 2D configuration with a geotube.