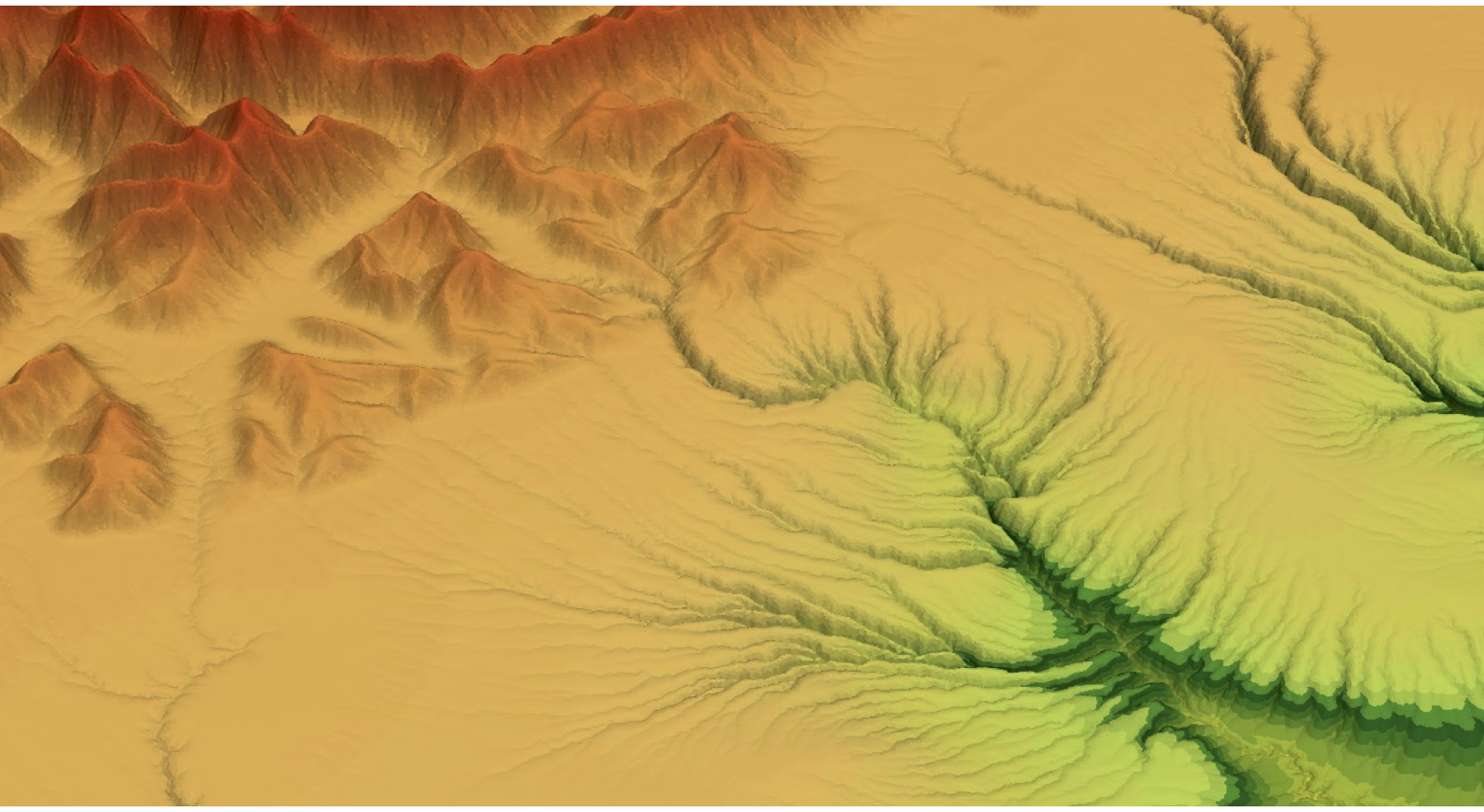
pyBadlands: User Documentation

November 2021



Tristan Salles, School of Geosciences, University of Sydney

Simulated processes

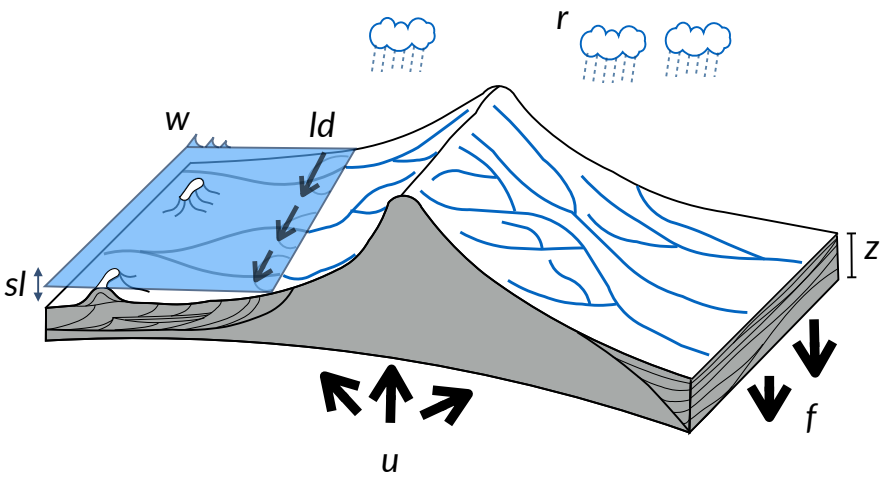


Fig 1.: A schematic of our model illustrating the main variables and forcing conditions. **w** represents the wave boundary conditions, **ld** the longshore drift, **sl** the sea-level, **u** the tectonic, **f** the flexural isostasy and **r** the rainfall patterns. The stratigraphic evolution and morphology are computed through time.

Over the last decades, many numerical models have been proposed to simulate how the Earth surface has evolved over geological time scales in response to different driving mechanisms such as tectonics or climatic variability [1, 2, 3, 4, 5]. These models combine empirical data and conceptual methods into a set of mathematical equations that can be used to reconstruct landscape evolution and associated sediment fluxes [6, 7]. They are currently used in many research fields such as hydrology, soil erosion, hillslope stability and general landscape studies.

Numerous models have focused on the erosional processes especially in the case of river dynamics [8, 9, 10, 11, 7] yet much less work has been done to simulate regional to continental depositional systems and associated stratigraphic evolution in sedimentary basins [6, 12]. In addition and with a few exceptions [13, 14, 15], most of these models have either been limited to one part of the sediment routing system (e.g., fluvial geomorphology, coastal erosion, carbonate platform development) or built upon simple laws commonly derived from diffusion-based equations which applications require pluri-kilometric spatial resolution [16]. These limitations often restrict our understanding of sediment fate from source to sink. It also makes it difficult to link site-specific observations to numerical model outputs.

pyBadlands framework and the development efforts behind it are intended to address these shortcomings. It provides a more direct and flexible description of the inter-connectivities between land, marine and reef environments, in that it explicitly links these systems together (Fig 1). **pyBadlands** is a python-friendly version of Badlands [17, 12] which provides a programmable and flexible front end to all the previous functionality of this former code and add new capability in regards to simulated processes, portability and usability. As mentioned above, the focus of the framework is primarily on the description of Earth surface evolution and sedimentary basins formation over regional to continental scale and geological time (thousands to millions of years). **pyBadlands**, introduced herein, is view as an integrative framework which provides a simple and adaptable numerical tool to explore Earth surface dynamics and quantify the feedback mechanisms between climate, tectonics, erosion and sedimentation.

pyBadlands uses a triangular irregular network (TIN) to solve the geomorphic equations presented below [18]. The continuity equation is defined using a finite volume approach and relies on the method described in Tucker et al. [19]. Fig 2 illustrates the relationships between the different components of the framework.

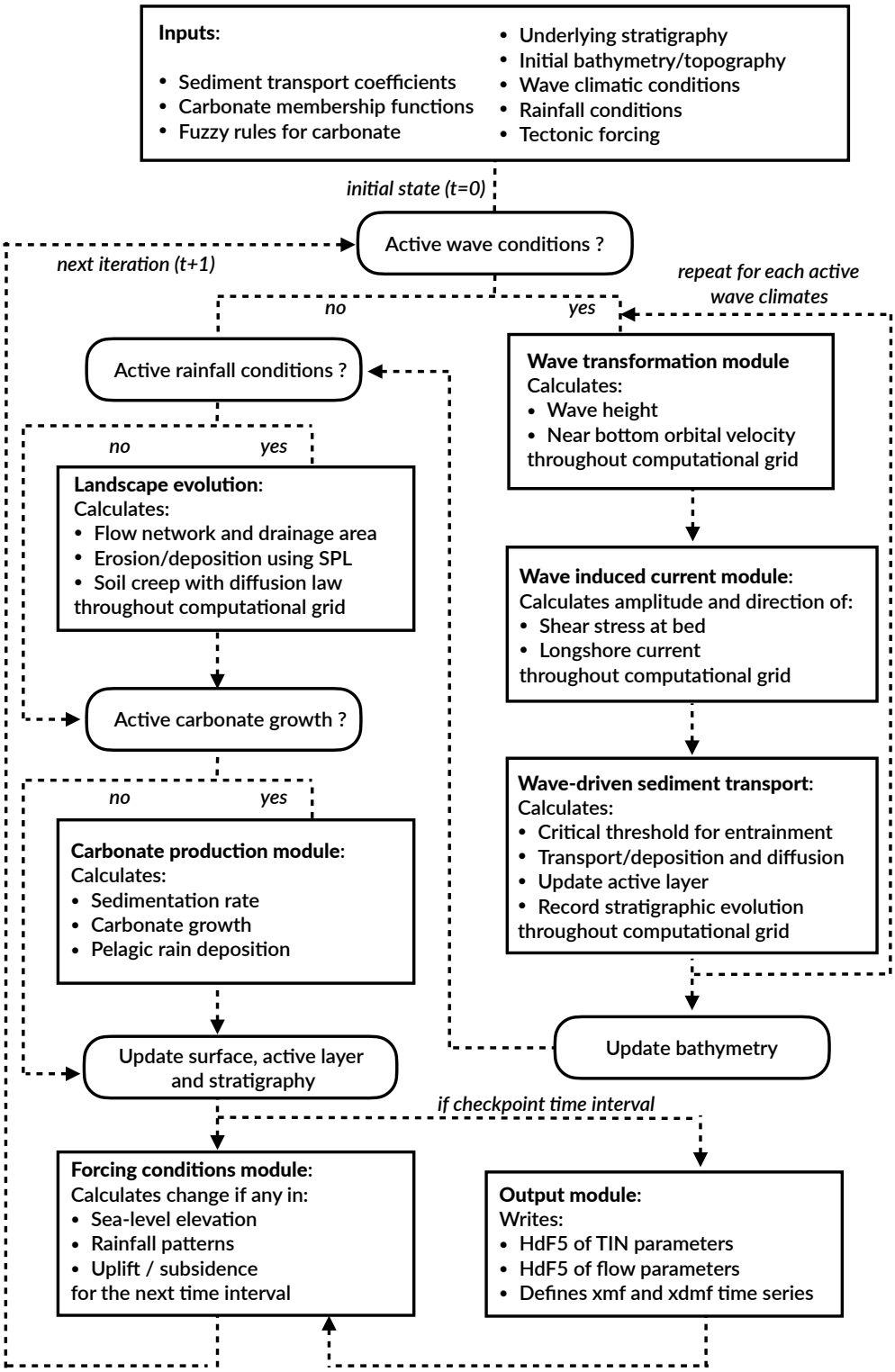


Fig 2.: Main components of **pyBadlands** workflow.

Fluvial system

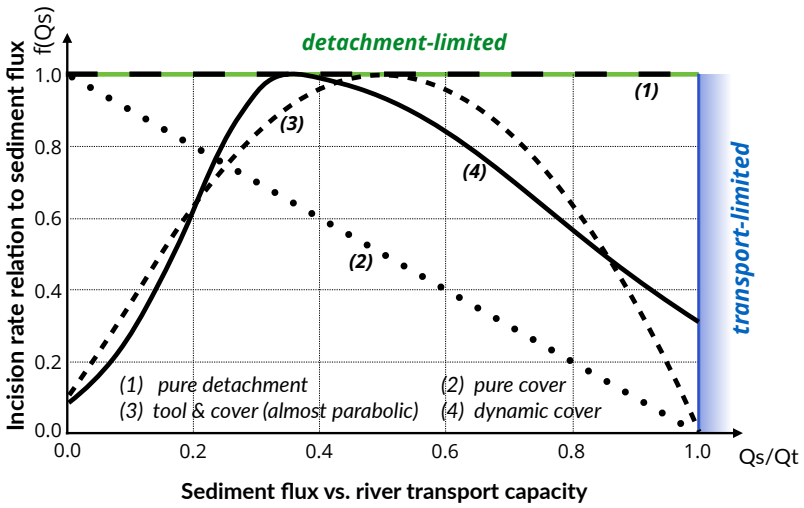


Fig 3.: Model space for stream power-based incision laws. It shows the dependence of river incision rate on sediment flux (adapted from Hobley et al. [7]).

To solve channel incision and landscape evolution, the algorithm follows the O(n)-efficient ordering method from Braun and Willett [20] and is based on a single-flow-direction (SFD) approximation assuming that water goes down the path of the steepest slope [21].

Several formulations of river incision have been proposed to account for long term evolution of fluvial system [2, 22]. These formulations describe different erosional behaviours ranging from detachment limited, governed by bed resistance to erosion, to transport limited, governed by flow capacity to transport sediment available on the bed. Mathematical representation of erosion process in these formulations is often assumed to follow a stream power law [15]. These relatively simple approaches have two main advantages. First, they have been shown to approximate the first order kinematics of landscape evolution across geologically relevant timescales ($>10^4$ years). Second, neither the details of long term catchment hydrology nor the complexity of sediment mobilisation dynamics are required. Below, we present the six incision laws currently available in pyBadlands (Fig 3).

Detachment-limited model

The default fluvial incision is based on the detachment-limited stream power law (Fig 3 - option 1), in which erosion rate $\dot{\epsilon}$ depends on drainage area A, net precipitation P and local slope S and takes the form:

$$\dot{\epsilon} = \kappa_d P^l (PA)^m S^n \tag{1}$$

κ_d is a dimensional coefficient describing the erodibility of the channel bed as a function of rock strength, bed roughness and climate, l, m and n are dimensionless positive constants. Default formulation assumes $l = 0$, $m = 0.5$ and $n = 1$. The precipitation exponent l allows for representation of climate-dependent chemical weathering of river bed across non-uniform rainfall [23].

Transport-limited models

Here, volumetric sediment transport capacity (Q_t) is defined using a power law function of unit stream power:

$$Q_t = \kappa_t (PA)^{m_t} S^{n_t} \tag{2}$$

where κ_t is a dimensional coefficient describing the transportability of channel sediment and m_t and n_t are dimensionless positive constants. In Eq 2, the threshold of motion (the critical shear stress) is assumed to be negligible [24].

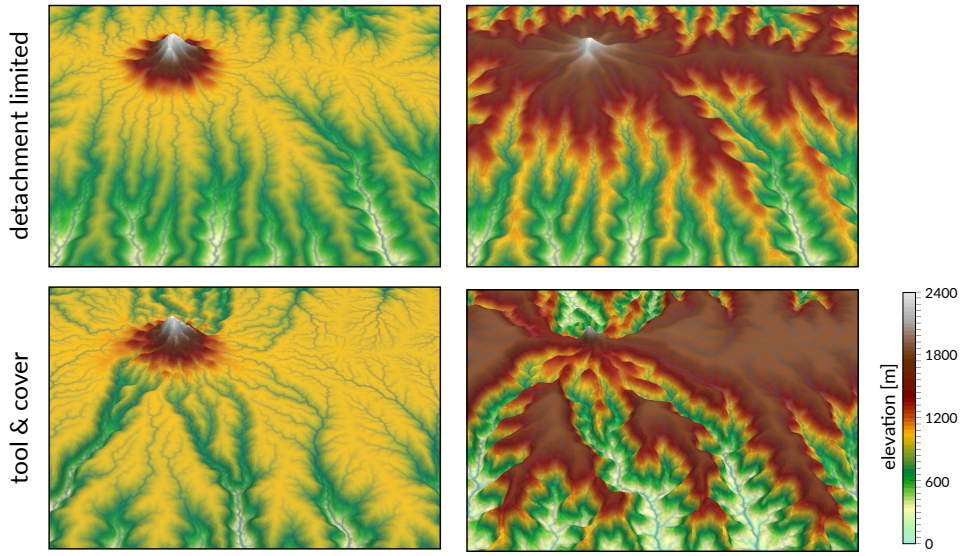


Fig 4.: Illustration of the impact of detachment versus transport limited (tool & cover option) formulations on landscape dynamics. The surface shows the evolution of the drainage system of an uplifted surface composed of an island and a plateau at 5 and 9 Ma

An additional term is now introduced in the stream power model:

$$\dot{\epsilon} = \kappa f(Q_s)(PA)^m S^n \tag{3}$$

with $f(Q_s)$ representing a variety of plausible models for the dependence of river incision rate on sediment flux (Fig 3). In the standard detachment-limited, $f(Q_s)$ is equal to unity, which corresponds to cases where $Q_s \ll Q_t$. Where sediment flux equals or exceeds transport capacity ($Q_s/Q_t \geq 1$) the system is by definition transport-limited (and depositional if $Q_s/Q_t > 1$).

Linear decline This model assumes that stream incision potential decreases linearly from a maximum where there is negligible sediment flux to zero where sediment flux equals transport capacity (Fig 3 - option 2). Conceptually, this law mimics the transfer of stream energy from erosion to transport processes [1]:

$$f(Q_s) = 1 - \frac{Q_s}{Q_t} \tag{4}$$

Almost parabolic Both qualitative and experimental observations have shown that sediment flux has a dual role in the erosion process. First, when sediment flux is low relative to capacity, erosion potential increases with sediment flux (tool effect: bedrock abrasion and plucking). Then, with increased sediment flux, erosion is inhibited (cover effect: sediments protect the bed from impacts by saltating particles). Following Gasparini et al. [25], we adopt a parabolic form that reaches a maximum at $Q_s/Q_t = 1/2$ [26, 27] (Fig 3 - option 3, Fig 4):

$$\begin{aligned} f(Q_s) &= 1 - 4\left(\frac{Q_s}{Q_t} - 0.5\right)^2 \quad \text{if} \quad Q_s/Q_t > 0.1, \\ f(Q_s) &= 2.6\frac{Q_s}{Q_t} + 0.1 \quad \text{if} \quad Q_s/Q_t \leq 0.1 \end{aligned} \tag{5}$$

Dynamic cover To account for sediment and spatial heterogeneity of the armouring of the bed, Turowski et al. [11] proposed a modified form of the 'almost parabolic' model that better estimates the original Sklar and Dietrich [10] experiments (Fig 3 - option 4). It takes the form of two combined Gaussian functions:

$$f(Q_s) = \exp\left[-\left(\frac{Q_s/Q_t - 0.35}{C_h}\right)^2\right] \tag{6}$$

where C_h is set to 0.22 for $Q_s/Q_t \leq 0.35$ and 0.6 when $Q_s/Q_t > 0.35$.

Saltation abrasion Sklar & Dietrich [26, 28] proposed also a formulation for tool and cover mechanisms which relates bedrock incision to abrasion of saltating bed load. The expression shares the same form as the sediment flux–dependent incision rule presented by Whipple & Tucker [1] (Eq 3) with significantly different exponent values:

$$\dot{\epsilon} = \kappa_{sa} f(Q_s) A^{-0.25} S^{-0.5} \tag{7}$$

in which the dependence of incision rate to sediment flux is defined as:

$$f(Q_s) = \frac{Q_s}{W}\left(1 - \frac{Q_s}{Q_t}\right) \tag{8}$$

where W represents the channel width and is expressed as a power law relation between width and discharge $W = \kappa_w A^b$.

Abrasion incision Parker [29] presented an incision model in which two processes dominate erosion of the channel bed: plucking of bedrock blocks and abrasion by saltating bed load. The approach takes the following form:

$$\dot{\epsilon} = \kappa_{ai} \frac{Q_s}{W}\left(1 - \frac{Q_s}{Q_t}\right) \tag{9}$$

where the only difference with the saltation-abrasion model is in the exponents on both the discharge and slope which are set to zero.

Hillslope processes

Along hillslopes, we assume that gravity is the main driver for transport and state that the flux of sediment is proportional to the gradient of topography (Fig 5).

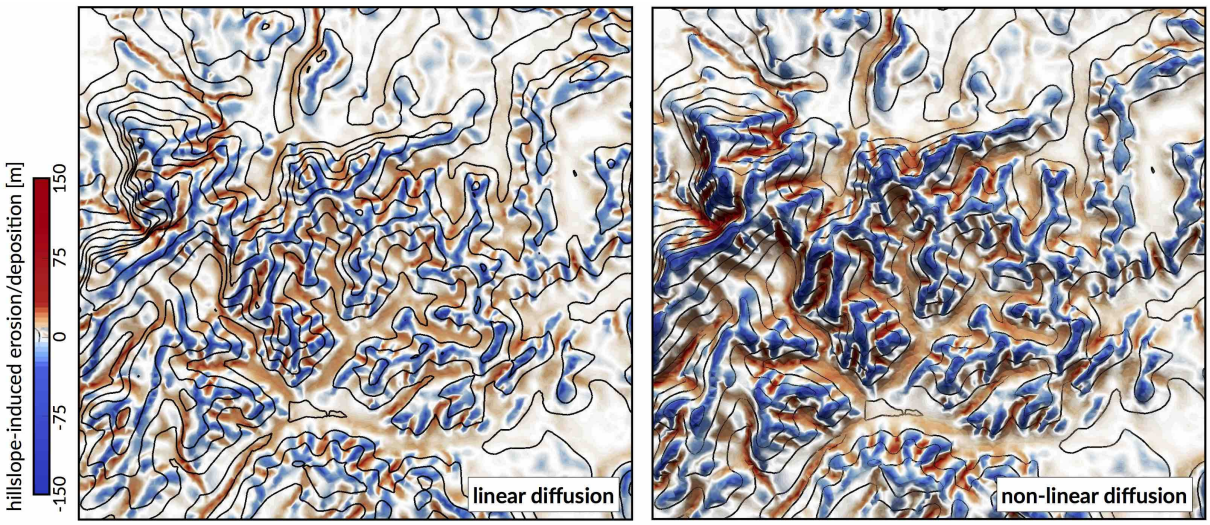


Fig 5.: Erosion/deposition induced after 130 ka of hillslope processes using the linear and non-linear formulations. Left: Linear diffusion produces convex upward slopes ($\kappa_{hl} = \kappa_{hn} = 0.05$). Right: non-linear approach tends to have convex to planar profiles as hillslope processes dominate when slopes approach or exceed the critical slope ($S_c = 0.8$) [30, 31]

One can choose between 2 options to simulate these processes. In the first, we use a linear diffusion law commonly referred to as soil creep [32, 33]:

$$\frac{\partial z}{\partial t} = \kappa_{hl} \nabla^2 z \tag{10}$$

in which κ_{hl} is the diffusion coefficient and can be defined with different values for the marine and land environments. It encapsulates, in a simple formulation, processes operating on superficial sedimentary layers. Main controls on

variations of κ_{hl} include substrate, lithology, soil depth, climate and biological activity.

Field evidence, however, suggests that the linear diffusion approximation (Eq 10) is only rarely appropriate [34, 35, 30, 31]. Instead, Andrews and Bucknam [36] and Roering et al. [37, 38] proposed a non-linear formulation of diffusive hillslope transport, assuming that flux rates increase to infinity if slope values approach a critical slope S_c. This alternative formulation is available as a second option and takes the following form in our model:

$$\frac{\partial z}{\partial t} = \nabla \cdot \frac{\kappa_{hn} \nabla z}{1 - (|\nabla z|/S_c)^2} \tag{11}$$

Wave-induced longshore drift – beta testing

We adopt the most basic known principles of wave motion, *i.e.*, the linear wave theory[39, 40]. Wave celerity c is governed by:

$$c = \sqrt{\frac{g}{\kappa} \tanh \kappa d} \tag{12}$$

where g is the gravitational acceleration, κ the radian wave number (equal to $2\pi/L$, with L the wave length), and d is the water depth. In deep water, the celerity is dependent only on wave length ($\sqrt{gL/2\pi}$); in shallow water, it depends on depth ($\sqrt{gd}$). From wave celerity and wave length, we calculate wave front propagation (including refraction) using the Huygens principle[41]. From this, we deduce the wave travel time and define wave directions from lines perpendicular to the wave front. Wave height (H) is then calculated along wave front propagation. The algorithm takes into account wave energy dissipation in shallow environment as well as wave-breaking conditions.
Wave-induced sediment transport is related to the maximum bottom wave-orbital velocity $u_{w,b}$. Assuming the linear shallow water approximation [42], its expression is simplified as:

$$u_{w,b} = (H/2)\sqrt{g/d} \tag{13}$$

Under pure waves (*i.e.*, no superimposed current), the wave-induced bed shear stress τ_w is typically defined as a quadratic bottom friction[43]:

$$\tau_w = \frac{1}{2} \rho f_w u_{w,b}^2 \tag{14}$$

with ρ the water density and f_w is the wave friction factor. Considering that the wave friction factor is only dependent of the bed roughness k_b relative to the wave-orbital semiexcursion at the bed A_b [44], we define:

$$f_w = 1.39 \, (A_b/k_b)^{-0.52} \tag{15}$$

where $A_b = u_{w,b} T/2\pi$ and $k_b = 2\pi d_{50}/12$, with d_{50} median sediment grain-size at the bed and T the wave period. For each wave condition, the wave transformation model computes wave characteristics and the induced bottom shear stress. These parameters are subsequently used to evaluate the long-term sediment transport active over the simulated region.

We assume that flow circulation is mainly driven by waves and other processes such as coastal upwelling, tidal, ocean or wind-driven currents are ignored. The proposed method consists in producing *snapshots* of wave-driven circulation distribution resulting from series of deep-water wave scenarios by computing time-averaged longshore current. In nearshore environments, longshore current runs parallel to the shore and significantly contributes to sediment transport[45, 46]. The longshore current velocity ($\vec{v_l}$) in the middle of the breaking zone is defined by [47]:

$$\vec{v_l} = \kappa_l u_{w,b} cos(\theta) sin(\theta) \vec{k} \tag{16}$$

with θ the angle of incidence of the incoming waves, κ_l a scaling parameter and $\vec{k}$ the unit vector parallel to the breaking depth contour.
In regions where wave-induced shear stress (Eq 14) is greater than the critical shear stress[48] derived from the Shields parameter ($\tau_c = \theta_c g d_{50} (\rho_s - \rho_w)$), bed sediments are entrained. The erosion thickness h_e is limited to the top sedimentary layer and for simplicity is assumed to follow a logarithmic form [49]:

$$h_e = C_e \ln(\tau_w/\tau_c) \text{ where } \tau_w > \tau_c \tag{17}$$

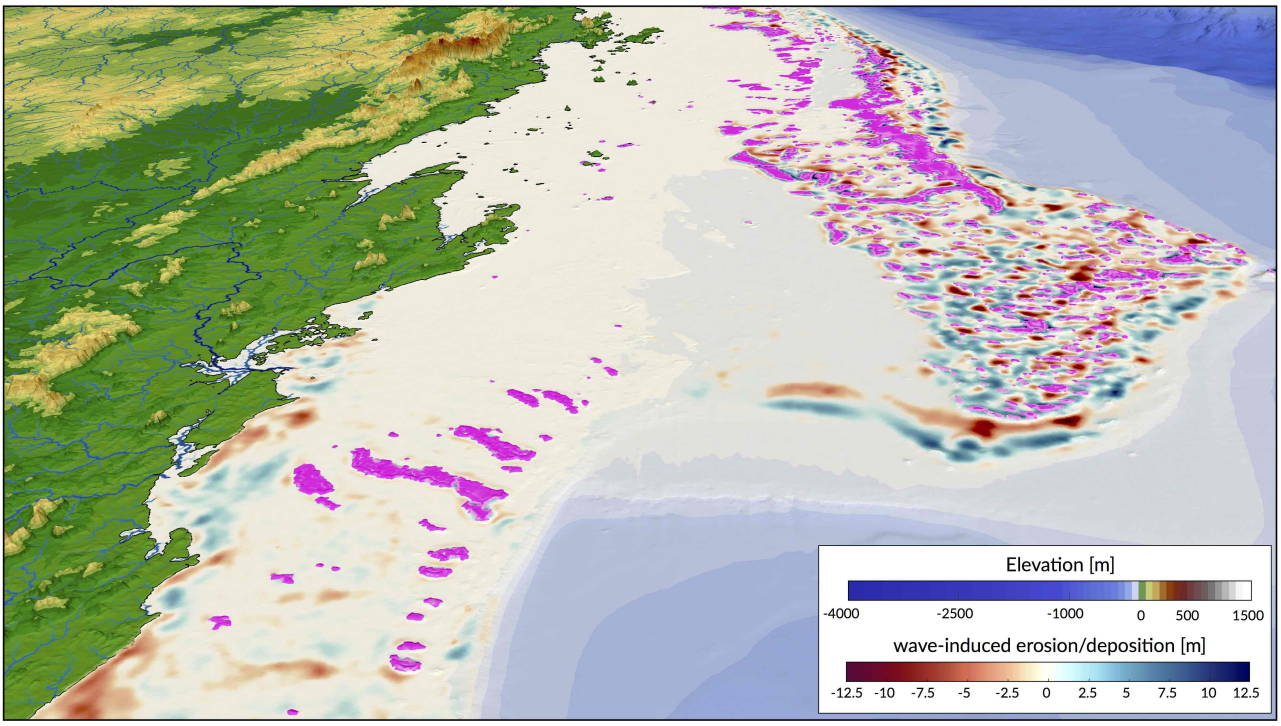


Fig 6.: Example of cumulative wave-induced erosion/deposition during a transgression event (simulated time: 14 ka). Wave-induced shear stress and associated longshore sediment transport are evaluated every 50 years. Pink patches show location of produced coral reefs.

where C_e is an entrainment coefficient controlling the relationship between shear stress and erosion rate[50]. Once entrained, sediments are transported following the direction of longshore currents and are deposited in regions where $\tau_w < \tau_c$[51] (Fig 6).

Carbonate production – beta testing

The organisation of coral reef systems is known to be large and complex and we are still limited in our understanding of their temporal and spatial evolution [52]. Additionally, most datasets of carbonate systems are often linguistic, context-dependent, and based on measurements with large uncertainties. Alternative approaches, such as fuzzy logic or cellular automata algorithms, have proven to be viable options to simulate these types of system [53, 54, 55]. Fuzzy logic methods are able to create logical propositions from qualitative data by using linguistic logic rules and *fuzzy sets* [56]. These fuzzy sets are defined with continuous boundaries rather than *crisp* discontinuous ones often used in conventional methods [57].

Based on a fuzzy logic approach, carbonate system evolution in pyBadlands is driven entirely by a set of rules whose variables are fully adjustable. The utility and effectiveness of the approach is mostly based on the user's understanding of the modelled carbonate system[14]. The technique is specifically useful to understand how particular variable, in isolation or in combination with other factors, influences carbonate depositional geometries and reef adaptation (Fig 7).

Here, carbonate growth depends on three types of control variables: depth (or accommodation space), wave energy (derived from ocean bottom orbital velocity) and sedimentation rate. For each of these variables, one can define a range of fuzzy sets using membership functions[56]. A membership function is a curve showing the degree of truth (*i.e.* ranging between 0 and 1) of membership in a particular fuzzy set (Fig 7). These curves can be simple triangles, trapezoids, bell-shaped curves, or have more complicated shapes as shown in Fig 7.

Production of any specific coral assemblage is then computed from a series of fuzzy rules. A fuzzy rule is a logic *if-then* rule defined from the fuzzy sets[53]. Here, the combination of the fuzzy sets in each fuzzy rule is restricted to

the *and* operator. The amalgamation of competing fuzzy rules is usually referred to as a fuzzy system. Summation of multiple rules from the fuzzy system by truncation of the membership functions produces a fuzzy answer in the form of a membership set (Fig 7). The last step consists in computing a single number for this fuzzy set through *defuzzification* [58]. In our approach, the centroid (centre of gravity) for the area below the membership set is taken as the *defuzzified* output value. The process returns a *crisp* value of coral assemblage growth on each cell of the simulated region (Fig 6).

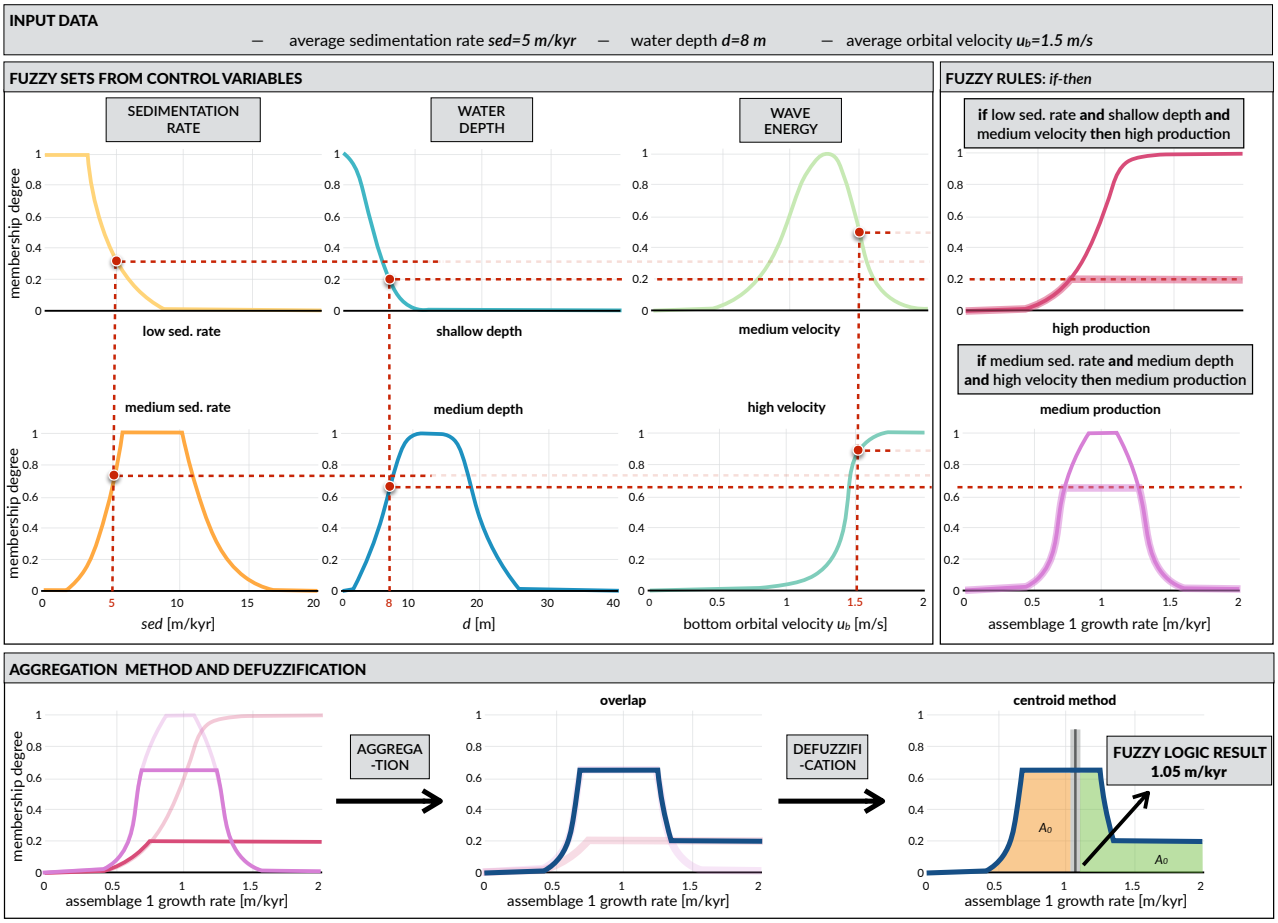


Fig 7.: Diagram of fuzzy logic process used to evaluate a specific coral assemblage growth rate

Extrinsic forcings

Over geological time scales, sediment transport from source to sink is primarily controlled by climate, tectonics and eustatism (Fig 1). In pyBadlands, the following set of external forcing mechanisms could be considered:

- sea-level fluctuations,
- subsidence, uplift and horizontal displacements,
- rainfall regimes and
- boundary wave conditions.

Tectonic forcing is driven by series of temporal maps. Each map can have variable spatial cumulative displacements making it possible to simulate complex 3D tectonic evolution with both vertical (uplift and subsidence) and horizontal directions. When 3D deformations is imposed, the model uses the node refinement technique proposed by Thieulot et al. [59]. The geometry of the surface is first advected by tectonic forces before being modified by surface processes. Due to tectonic advection, the density of the nodes evolves over time, which could lead to unbalanced resolutions with places showing either rarefaction or accumulation of nodes. To limit this effect, the advected surface

is modified by adding or removing nodes to ensure homogeneous nodes distribution.

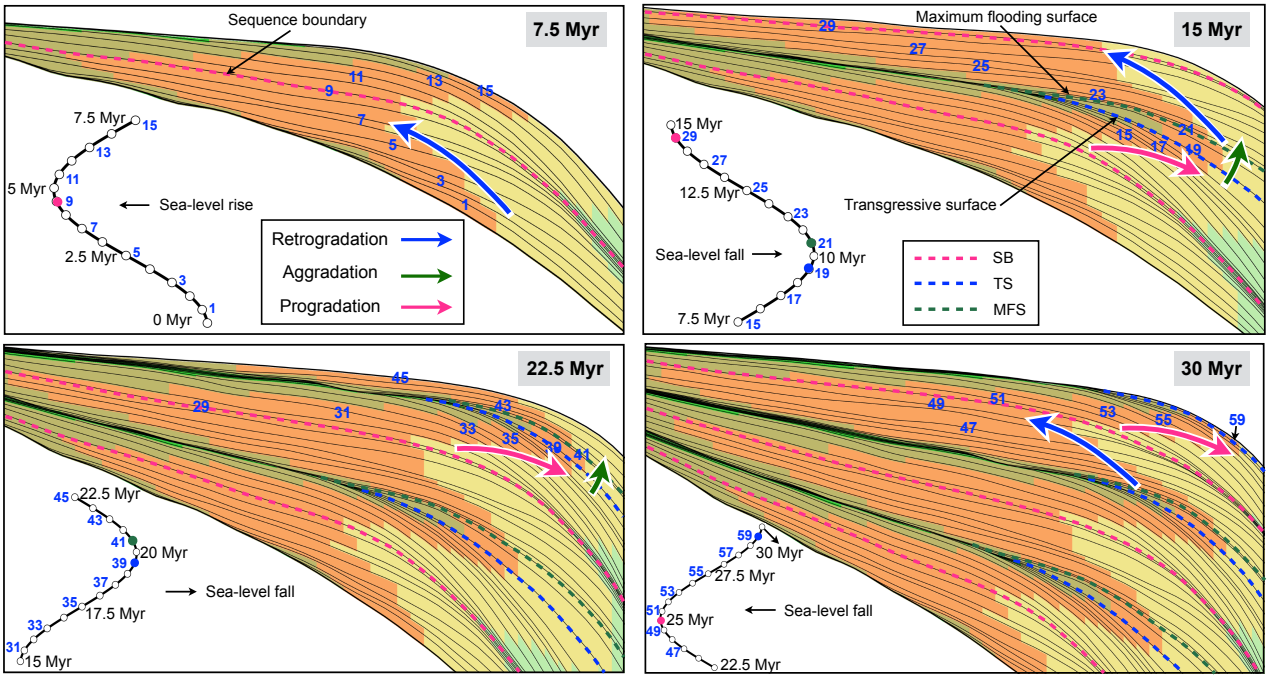
Sea-level curves (sea-level elevation through time) can be imposed from either a published eustatic curve [60, 61] or directly defined by the user.

Temporal variations in precipitation might be applied either as a constant values (metres per year) or a set of maps representing spatially changing rainfall regimes. In addition, to account for the interactions between rainfall and topography, the option is given to use the orographic precipitation linear model from Smith & Barstad [62]. As an example, the coupled evolution of precipitation patterns and topography can be use to quantify the relative importance of climate, erosion and tectonic in mountain geomorphology.

Wave conditions To evaluate marine sediment transport over several thousands of years, the approach taken here relies on stationary representation of prevailing fair-weather wave conditions. The wave transformation model is generally performed for time intervals varying from 5 to 50 years. The aim is to simulate realistic wave fields by imposing a sequence of wave forcing conditions. At any given time interval, we define a percentage of activity for each deep-water wave conditions as well as the significant wave height. Then, the bathymetry is used to compute associated wave parameters.

The forcing mechanisms described above will directly control the evolution of sediment transport, associated stratal architecture as well as carbonate production.

Usability



Portability

pyBadlands is an open-source package distributed under GNU GPLv3 license. The source code is available on GitHub. Structurally the code is a python front end with a C and Fortran middle layer to efficiently compute some of the heaviest functions. This python-friendly version provides a programmable and flexible interface which maximises its portability across platforms.

Instructions to install the code and the associated dependencies on a local system are provided on our wiki pages along with model options and a series of hands-on examples.

The easiest way to use **pyBadlands** is via our Docker container (searching for pybadlands-demo-dev on Kitematic) which is shipped with the complete list of dependencies, the model companion and the examples. Models data and outputs ran from within the container will not persist when that container is no longer active. To provide better interfacing between the container and the host filesystem, **pyBadlands** image can be mounted on a local volume which allow for easy access and ability to store securely model results.

Interactions with other packages

pyBadlands main calculations are performed on a TIN. However the code creates its own Delaunay triangulation using the Shewchuk's Triangle library [63] from regularly defined input datasets (e.g., topography grid, rain maps, tectonic maps). In that way, users provide standard regularly spaced ASCII datasets. The only requirement is to follow a specific column-major order for the declaration of each nodes values which is consistent between imported datasets starting from the south-west and ending on the north-east corner.

Exp.	dim. [km]	res.	time	Forcings & Processes			
Basin	30 x 30	100 m	1 Ma	Rain uni.	s.l.	Fluv./hillslp.	Strat.
Crater	2.5 x 2.5	10 m	200 ka	Rain uni.		Fluv./hillslp.	
Delta	25 x 25	50 m	500 ka	Rain uni.	s.l./tect.	Fluv./hillslp.	Strat.
Dyntopo	300 x 200	1 km	10 Ma	Rain uni.	s.l.	Fluv./hillslp.	
eTopo	133 x 180	50 m	500 ka	Rain uni.	s.l./tect.	Fluv./hillslp.	
Flexure	250 x 100	500 m	1 Ma	Rain uni.	gflex	Fluv./hillslp.	
Mountain	80 x 40	400 m	10 Ma	Rain oro.		Fluv./hillslp.	
Rift	400 x 400	2 km	250 ka	Rain uni.	3D tect.	Fluv./hillslp.	
Strikeslip	200 x 200	1 km	100 ka	Rain uni.	3D tect.	Fluv./hillslp.	

Table 1.: Summary of hands-on examples provided with **pyBadlands** package. Abbreviations: dim.: model dimension – rain uni./oro.: uniform or orographic – res.: model resolution – fluv.: fluvial processes – hillslp.: hillslope – strat.: stratigraphic architecture – s.l.: sea-level – tect.: tectonics.

Model results consist of time series of surface evolution, river and catchment dynamics grids as well as underlying stratigraphic architecture mesh. These outputs are all produced as Hdf5 binary files making it possible to interact with multiple existing visualisation and analysis software, such as *Paraview* or *MayaVi*.

Initial surface can be generated from UTM coordinates and functions have been added to easily extract Web Map Service dataset (one example is provided to illustrate how to define an initial topography grid from ETOPO5 datasets). **Hdf5** files can also be quickly transformed in other conventional raster GIS file formats such as ASCII grids.

To estimate flexural isostasy, **gFlex** modular python package [64] has been integrated as a component in pyBadlands. It allows to compute isostatic deflections of Earth's lithosphere with uniform or nonuniform flexural rigidity and couple the interactions with evolving surface loads induced by erosion/deposition associated to modelled surface processes.

Hands-on examples

A series of examples are available with the source code. They illustrate the different capabilities of our package and are an ideal starting point to learn how to use **pyBadlands**. Each folder is composed of

- an input XML file where the different options for the considered experiment are set,
- a *data* folder containing the initial surface and potentially some forcing files (e.g. sea-level, rainfall or tectonic grids) and
- a series of IPython notebooks used to run the experiment and perform some pre or post-processing tasks.

These examples have been designed to be run quickly and should take on average 5 minutes on standard computer. The range of simulations varies both in term of spatial and temporal resolutions. A summary of the proposed models is presented in table 1 and could serve as a basis for more complex problems. You can browse the list of examples directly from the IPython notebooks.

Companion

To assist users during the pre and post-processing phases, a series of Python classes are proposed in a GitHub pyBadlands-Companion repository. These classes are shipped with the Docker container mentioned in previous section. In addition, IPython notebooks have been created to illustrate how these python classes are used. We have chosen this structure to give users the transparency and opportunity to

1. clearly understand the creation and format of the input files,
2. perform quantitative analyses of **pyBadlands** output files,
3. easily design their own notebooks and further improve the proposed workflow.

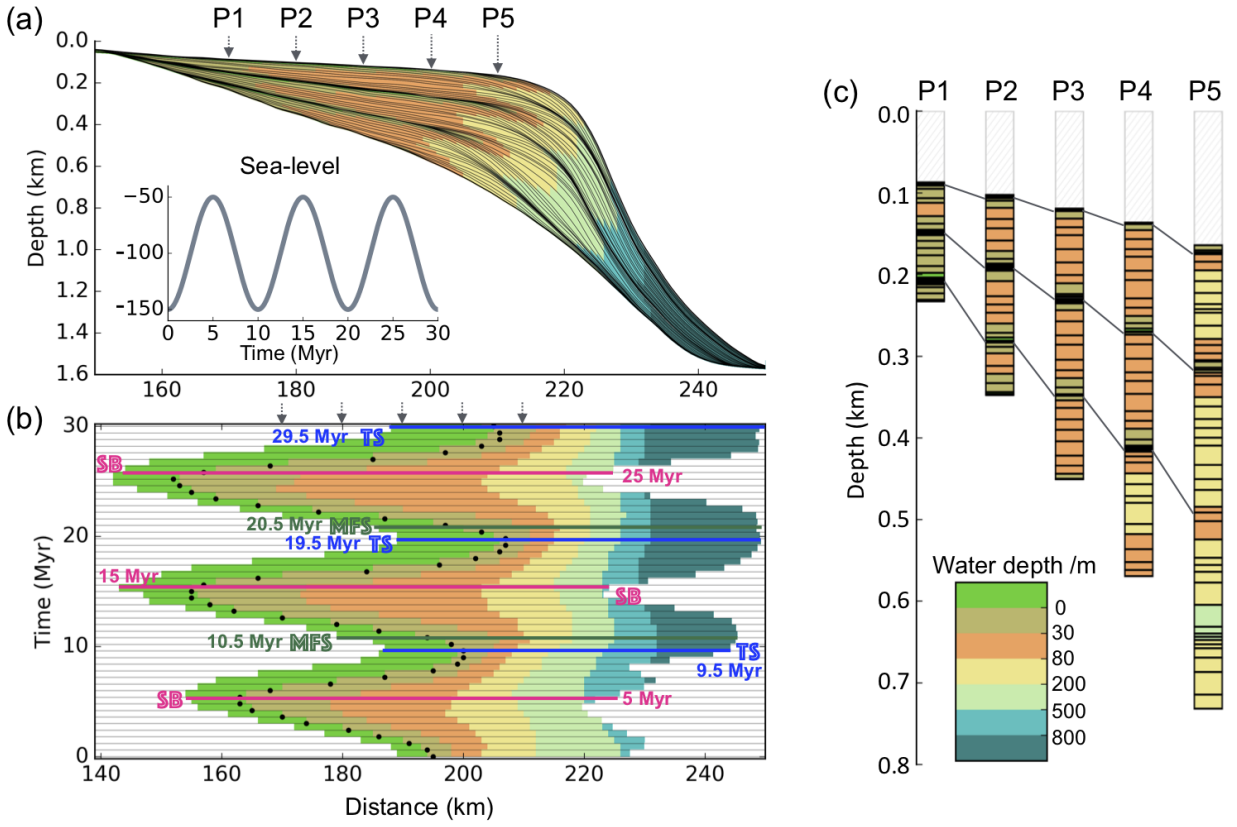


Fig 1.: Predicted stratal architecture. (a) Stratal stacking patterns on a vertical cross-section. Solid black lines shown on each subplot are stratigraphic layers and are plotted at 0.5 Myr intervals. The coupled sea-level scenario is modelled as a sinusoidal curve. Different colours stand for different depositional environments that are defined based on water depth. (b) Wheeler diagram or chronostratigraphy chart. The black dots are shoreline positions through time, or shoreline trajectory. The coloured dashed lines are stratigraphic surfaces identified based on stratal terminations, stacking trends and shoreline trajectory. (c) Virtual cores extracted at different positions: P1, P2, P3, P4, P5. Solid lines connect condensed sections that are associated with unconformity due to sea-level fall.

Pre-processing classes

The pre-processing notebooks allows for quick creation of grids and files compatible with pyBadlands input format. The main functionalities and associated notebook filenames are listed below:

- topographic grids for generic model (topoCreate),
- real topographic/bathymetric dataset (etopoGen),
- building sea level fluctuations curve or using Haq curve [60] (seaLevel),
- horizontal displacement and precipitation maps (topoTec),
- regridding initial tectonic, rainfall and topographic input files (regridInput)

Post-processing classes

Morphometric & Hydrometric The morphometrics notebook can be used to perform quantitative analyses of simulated pyBadlands landforms [3, 12]. Gradients, curvature (horizontal and vertical), aspect and discharge attributes can be extracted for the entire region or a specific area of the simulation. The hydrometric notebook allows for evaluation of time dependent evolution of a specific catchment. It can be used to quantify the longitudinal evolution of a river profile, compute the Peclet number distribution, χ-maps as well as hypsometric curves.

Stratigraphy & Wheeler diagram When the stratigraphic structure is turned on in pyBadlands, it is possible to extract cross-section and plot stratigraphic layers (Fig 1a), Wheeler diagram (Fig 1b) and virtual cores (Fig 1c). The notebook extracts simulated depositional sequences on a vertical cross-section, and calculates the relative sea level change, shoreline trajectory, accommodation and sedimentation change (Fig 1). Three methods can be applied to interpret the stratigraphic units including

- the systems tracts model based on relative sea level change;
- the shoreline trajectory analysis [65]; and
- the accommodation succession method [66, 67].

Using the stratalMesh notebook, it is also possible to export the simulated stratigraphy as a VTK structured mesh that could be further analysed in other software packages.

XmL input file

General structure

In **pyBadlands**, a **XML** input file is used to set the parameters and conditions which apply to a given simulation. Here we present the complete list of parameters that can be used in the current version of the code.

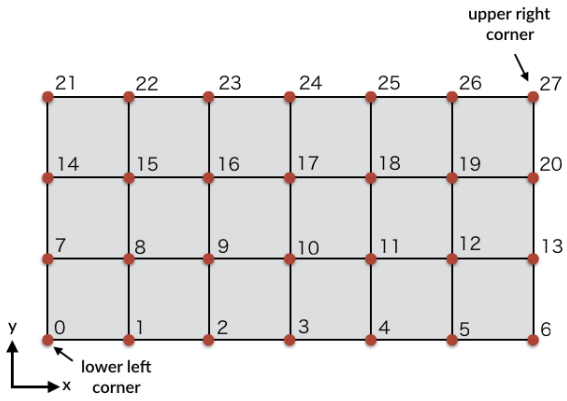

```
<?xml version="1.0" encoding="UTF-8"?>
<badlands xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

Grid structure

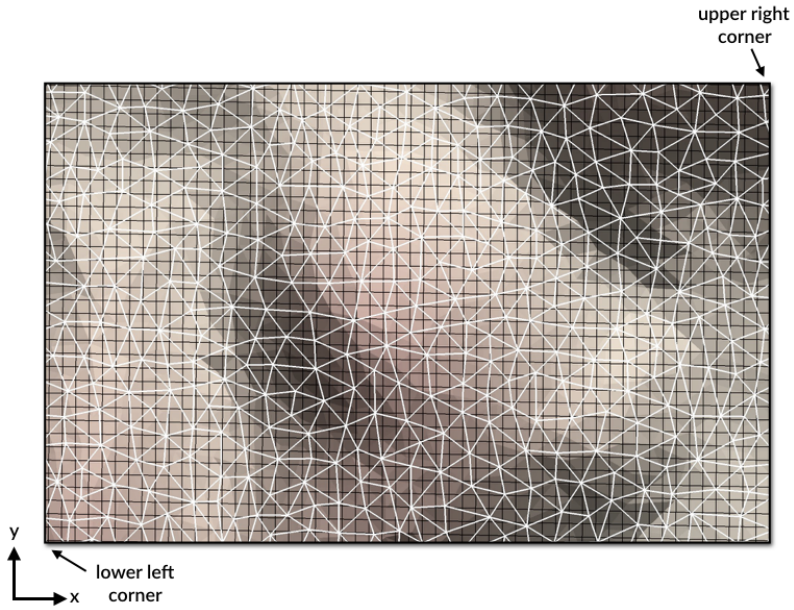
All **pyBadlands** runs require the definition of the *grid structure* that defines the initial surface over which surface processes are computed.

```
    <!-- Regular grid structure -->
    <grid>
        <!-- Digital elevation model file path -->
        <demfile>data/regularMR.csv</demfile>
        <!-- Optional parameter (integer) used to decrease TIN resolution.
             The default value is set to 1. Increasing the factor
             value will multiply the digital elevation model resolution
             accordingly.  -->
        <resfactor>2</resfactor>
        <!-- Boundary type: flat, slope, fix or wall -->
        <boundary>slope</boundary>
        <!-- Optional parameter (integer) used to force depression-less
             surface at the start of the simulation. The default value is 0
             to turn the option off, put it to 1 to enable it. (Optional) -->
        <nopit>0</nopit>
    </grid>
```

pyBadlands main calculations are performed on a triangular irregular network (TIN). However the code creates its own triangulation based on regularly defined dataset. In that way, the user is only required to provide some simple regularly spaced dataset. A similar approach is used for any other maps (e.g. rain, tectonics...).
The only requirement is to follow a specific order for the declaration of each nodes values. To create a simple surface or extract a surface from eTopo grid, you can follow the examples provided in the Companion (simple geometrical model and etopo1). The order starts from the lower left corner to the upper right corner based on a row-wise indexing (along the X-axis first) as shown in the figure below:



From the regular grid, the TIN is then created within **pyBadlands** with a resolution which is at maximum equal to the regular grid resolution provided by the user but which could be coarsened if the <**resfactor**> is set above one in the *grid structure*.



Time structure

The *time structure* is also a required element in the XML input and defines the duration of the simulation from **start** to **end** time (in years). It is worth noting that these times can be negative, for example <*start*> can be equal to -150,000 years and <*end*> to -75,000 years. The only requirement is that start time needs to be lower than end time... (don't forget this is a forward model!)

In cases where a model is restarted from a previous run, the <**restart**> element can be used and requires the previous model output folder and the step at which the new simulation needs to restart. **If the simulation is not restarting this element needs to be commented or deleted**.

By default when using the conventional stream power law incision, the model defines its own time step to ensure stability. The user however has the ability to control the minimum and maximum time step allowed.

Lastly, the user needs to define the display interval (<**display**>) which corresponds to the time step (in years) when an output is created. Depending of the size of your model, decreasing the number of output by increasing the display interval will make your simulation run quicker. When a stratigraphic mesh is recorded by the model (see next structure) the user can decide to output it not at every display interval but at any specific multiple of it. Again this may help to run a model faster.

```
<!-- Simulation time structure -->
<time>
    <!-- Restart structure -->
    <restart>
      <!-- Model output folder name to restart the simulation from -->
      <rfolder>output_01</rfolder>
      <!-- Model output file step number to restart the model from -->
      <rstep>3</rstep>
    </restart>
    <!-- Simulation start time [a] -->
    <start>0.</start>
    <!-- Simulation end time [a] -->
    <end>100000.</end>
    <!-- Minimum time step [a]. Default is 1. -->
    <mindt>1.</mindt>
    <!-- Maximum time step [a] (optional).
         Set to display interval is not provided. -->
    <maxdt>1000.</maxdt>
    <!-- Display interval [a] -->
    <display>5000.</display>
    <!-- Mesh output frequency based on the display interval. (integer)
         Considering a display interval of T yrs and a mesh output of K
         the mesh will be stored every T*K yrs - (optional default is 1) -->
    <meshout>28</meshout>
</time>
```

Stratal structure

```
<!-- Simulation stratigraphic structure -->
<strata>
    <!-- Stratal grid resolution [m] -->
    <stratdx>500.</stratdx>
    <!-- Stratal layer interval [a] -->
    <laytime>2500.</laytime>
    <!-- Surface porosity  -->
    <poro0>0.52</poro0>
    <!-- characteristic constant for Athy's porosity law  [/km] -->
    <poroC>0.47</poroC>
</strata>
```

This element is optional and needs to be loaded in cases where you want to record the stratigraphic architecture over time. It requires 2 parameters. First the horizontal resolution of the mesh that will be used to record the deposition thicknesses over time. This grid can have as a maximum the resolution of the topographic grid or can have a coarser resolution that needs to be a *factor* of the topographic grid resolution.

The second parameter is the time interval used to record the stratigraphic grid (<**laytime**>). It could be the same as the display interval or a smaller interval as long as it remains a multiple of it. For example, if your display interval is set to 25,000 laytime can for example be 25,000 or 12,500 or 5,000...

All clastic sediments are subject to compaction (and reduction of porosity) as the result of increasingly tighter packing of grains under a thickening overburden. Porosity decreases with depth, initially largely due to mechanical compaction of the sediment. The decrease in porosity is relatively large close to the seafloor, where sediment is loosely packed; the lower the porosity, the less room there is for further compaction. This decrease in porosity with depth is commonly modeled as a negative exponential function (Athy, 1930). This is an empirical equation, as there is no direct physical link between depth and porosity; compaction and porosity reduction are more directly related to the increase in effective stress under a thicker overburden. Here we only address the simplest scenario with no

overpressured zones. For normally pressured sediments, Athy's porosity-depth relationship can be expressed in the form:

$$\phi(d) = \phi_0 e^{-cd} \tag{18}$$

where the porosity ϕ varies with depth (d) based on surface porosity ϕ_0 (defined in the XmL by the element (<**poro0**>) and c a coefficient with the units [km^{-1}] (<**poroC**>).

Sea-level structure

By default, the sea-level position in **pyBadlands** is set to 0 m. If you wish to set it to another position you can use the <**position**> parameter that changes the sea-level to a new value relative to sea-level. Another option consists in defining your own sea-level curve (<**curve**>) or using a published one (e.g. Haq curve for example). To create your own sea-level curve, one can use the following example from the Companion: toolSea class.

```
<!-- Sea-level structure -->
<sea>
    <!-- Relative sea-level position [m] -->
    <position>-103450.</position>
    <!-- Sea-level curve - (optional) -->
    <!--curve>data/sealvl.csv</curve-->
</sea>
```

The sea-level curve is defined as a 2 columns ASCII file containing in the first column the time in years (they don't need to be regularly temporally spaced) and in the second the sea-level position for the given time. When the model runs, it will interpolate linearly between the defined times to define the position of the sea-level.

Tectonic structure

```
<!-- Tectonic structure -->
<tectonic>
    <!-- Is 3D displacements on ? (1:on - 0:off). Default is 0.-->
    <disp3d>0</disp3d>
    <!-- Only relevant when 3D displacements is on.
         Closest distance [m] between nodes before
         merging happens. This is optional if not given
         the merging distance is set to half the resolution
         of the digital elevation input file. -->
    <merge3d>200.</merge3d>
    <!-- Only relevant when 3D displacements is required.
         This is useful if the horizontal displacements provided
         in each maps are larger than the TIN resolution. In this
         case, it is recommended to split each displacement periods
         in evenly spaced intervals of given time duration [a]. -->
    <time3d>5000.</time3d>
    <!-- Number of tectonic events -->
    <events>1</events>
    <!-- Displacement definition -->
    <disp>
        <!-- Displacement start time [a] -->
        <dstart>5.</dstart>
        <!-- Displacement end time [a] -->
        <dend>10.0</dend>
        <!-- Displacement map [m] -->
        <dfile>data/disp1D.csv</dfile>
    </disp>
</tectonic>
```

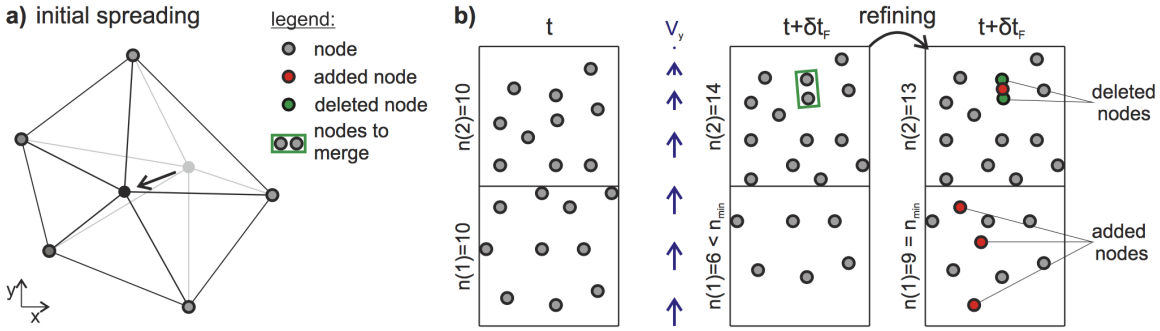
As for the sea-level structure, the tectonic one is optional. **pyBadlands** accepts both horizontal and vertical displacements. These displacements are either lithospheric or mantle induced. Actually the code does not care about what is inducing these changes.

Nevertheless the definition of vertical-only and horizontal+vertical displacements implies the declaration of different parameters.

In the most simple case of **vertical-only displacements** (i.e. uplift or subsidence) the model requires:

- the declaration of the element <**events**> that basically defines the number of tectonic fields to be applied during the simulation duration,
- the definition of each displacement event <**disp**>.

You will need to make sure that the number of events matches the number of displacements defined. Each displacement requires a start (<**dstart**>) and end (<**dend**>) time and a displacement map (<**dfile**>). The displacements map in the **vertical-only** case if defined as a ASCII file containing 1 column ordered in the same way as the topography file (lower left corner to upper right one based on row-wise indexing). The values of each node is set based on the desired displacement field and corresponds to the cumulative displacements during the given period.



The second, more complex, option (<**disp3d**>=1 – **horizontal+vertical displacements**) requires additional parameters. Due to tectonic advection, the density of the surface nodes evolves over time, which leads to areas showing *rarefaction* or *accumulation* of nodes. In order for the interpolation schemes to remain accurate and to avoid unnecessary computations, a local addition and deletion of nodes and the consequent remeshing of the triangulated surface are therefore required. This is done by defining the closest distance between nodes before merging happens (<**merge3d**>). The addition of points is done automatically based on the resolution of the initial topographic grid. To avoid unnecessary remeshing and prevents a huge distortion of the grid due to advection, user is required to set an internal time step for remeshing (<**time3d**>). Finally, the definition of the displacement file (<**dstart**>), in this case, requires the declaration of the cumulative displacements over the given period along the X, Y and Z directions. Thus this file has 3 columns (for each coordinates) and follows the same order as the topographic file. For an in-depth understanding of the technique, users need to look at the 3D surface deformations proposed by Thieulot et al., 2014.

Precipitation structure

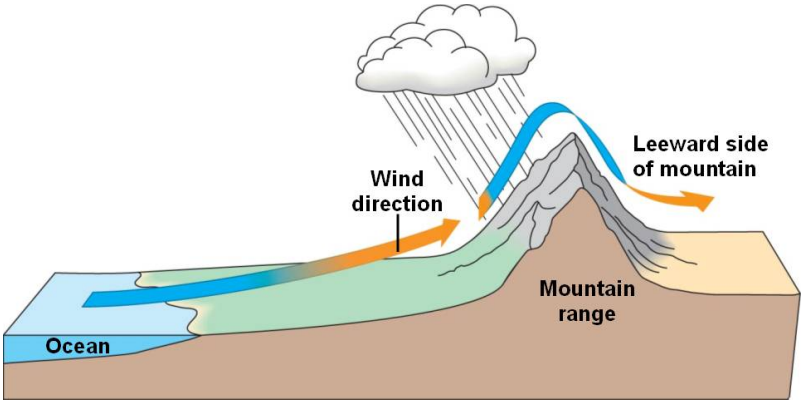
Except in cases where you are only interested in aerial evolution associated to hillslope processes only, you will need to define the precipation structure to account for fluvial related sediment transport.

Like for the tectonic structure, you will be able to define both spatial and temporal changes in the precipitation regime over the simulation time. The first element (<**climates**>) specifies the number of temporal rain event that you will impose. You will need to ensure that this number matches with the number of (<**rain**>) element that will be declared (otherwise the code will complain during execution).

```
<!-- Precipitation structure
     The following methods can be used:
        - an uniform precipitation value for the entire region [m/a]
        - a map containing precipitation values for each nodes of the regular
        - an orographic precipitation computed using Smith & Barstad theory (2
-->
<precipitation>
    <!-- Number of precipitation events -->
    <climates>3</climates>
    <!-- Uniform precipitation definition -->
    <rain>
        <!-- Rain start time [a] -->
        <rstart>0.</rstart>
        <!-- Rain end time [a] -->
        <rend>100000.</rend>
        <!-- Precipitation value [m/a] -->
        <rval>1.</rval>
    </rain>
    <!-- Precipitation map definition -->
    <rain>
        <!-- Rain start time [a] -->
        <rstart>100000.</rstart>
        <!-- Rain end time [a] -->
        <rend>200000.</rend>
        <!-- Precipitation map [m/a] -->
        <map>data/rainLR.csv</map>
    </rain>
```

The <**rain**> structure contains at least 3 parameters: the start and end time of the given event and the definition of the precipitation values in metres per year. For this last parameter, three methods are available:

1. an uniform precipitation value for the entire region (<**rval**>),
2. a precipitation map containing spatially varying values (<**map**>), again this map is defined as a ASCII file containing 1 column ordered in the same way as the topography file (lower left corner to upper right one based on row-wise indexing),
3. an orographic precipitation model which accounts for the change in rainfall patterns associated to change in topography. The orographic precipitation uses Smith & Barstad (2004) linear model to compute topographic induced rain field and potential values are provided in the example below.



```
        <!-- Orographic precipitation model definition -->
        <rain>
            <!-- Rain start time [a] -->
            <rstart>100000.</rstart>
            <!-- Rain end time [a] -->
            <rend>200000.</rend>
            <!-- Rain computation time step [a] -->
            <ortime>5000.</ortime>
            <!-- Background precipitation value [m/a] -->
            <rbgd>1.</rbgd>
            <!-- Minimal precipitation value [m/a] -->
            <rmin>0.2</rmin>
            <!-- Maximal precipitation value [m/a] -->
            <rmax>4.</rmax>
            <!-- Wind velocity along X (W-E) direction [m/s] -->
            <windx>-3.</windx>
            <!-- Wind velocity along Y (S-N) direction [m/s] -->
            <windy>2.</windy>
            <!-- Time conversion from cloud water to hydrometeors
                 range from 200 to 2000 [s]. Optional default is set
                 to 1000 s -->
            <tauc>1000.</tauc>
            <!-- Time for hydrometeor fallout range from 200 to 2000 [s].
                 Optional default is set to 1000 s -->
            <tauf>1000.</tauf>
            <!-- Moist stability frequency range from 0 to 0.01 [/s].
                 Optional default is set to 0.005 /s -->
            <nm>0.005</nm>
            <!-- Uplift sensitivity factor range from 0.001 to 0.02 [kg/m3].
                 Optional default is set to 0.005 kg/m3 -->
            <cw>0.005</cw>
            <!-- Depth of the moist layer range from 1000 to 5000 [m].
                 Optional default is set to 3000 m -->
            <hw>3000.</hw>
        </rain>
    </precipitation>
```

Surface processes structure

Several formulations of river incision have been implemented and describe different erosional behaviours ranging from detachment limited, governed by bed resistance to erosion, to transport limited, governed by flow capacity to transport sediment available on the bed.

The default law available in **pyBadlands** is based on the detachment-limited equation, in which erosion rate $\dot{\epsilon}$ depends on drainage area A, net precipitation P and local slope S and takes the form:

$$\dot{\epsilon} = \kappa_d (PA)^m S^n \tag{19}$$

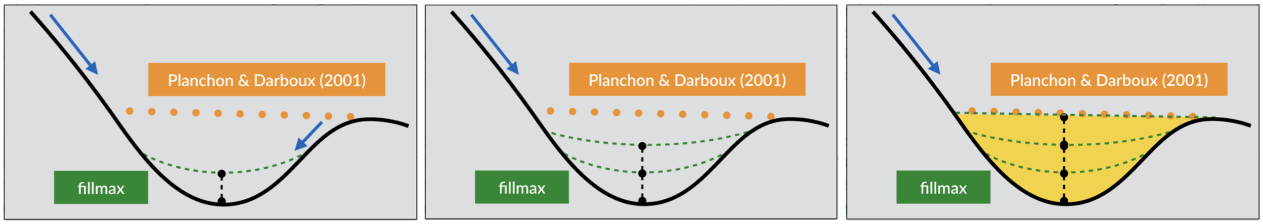
κ_d (defined as the <**erodibility**> element in the XmL) is a dimensional coefficient describing the erodibility of the channel bed as a function of rock strength, bed roughness and climate, m and n are dimensionless positive constants and are set using <**m**> and <**n**> respectively. Default formulation assumes $m = 0.5$ and $n = 1$.

This law is often used to look at purely erosive model (<**dep**>=0), but as our goal is to not only look at erosion but also at the evolution of sedimentary basin, we need to set additional parameters.

```
    <!--
         Sediment transport equation: stream power law model <sp_law>
    -->
    <!-- Stream power law parameters:
         The stream power law is a simplified form of the usual expression of
         sediment transport by water flow, in which the transport rate is assumed
         to be equal to the local carrying capacity, which is itself a function o
         boundary shear stress. -->
    <sp_law>
        <!-- Make the distinction between purely erosive models (0) and erosion /
            deposition ones (1). Default value is 1 -->
        <dep>1</dep>
        <!-- Maximum lake water filling thickness. This parameter is used
             to defined maximum water level in depression area.
             Default value is set to 200 m. -->
        <fillmax>200.</fillmax>
        <!-- Critical slope used to force aerial deposition for alluvial plain,
             in [m/m] (optional). -->
        <slp_cr>0.001</slp_cr>
        <!-- Maximum percentage of deposition at any given time interval from riv
             sedimentary load in alluvial plain. Value ranges between [0,1] (opti
        <perc_dep>0.5</perc_dep>
        <!-- Values of m and n indicate how the incision rate scales
             with bed shear stress for constant value of sediment flux
             and sediment transport capacity.
             Generally, m and n are both positive, and their ratio
             (m/n) is considered to be close to 0.5 -->
        <m>0.5</m>
        <n>1.0</n>
        <!-- The erodibility coefficient is scale-dependent and its value depend
              on lithology and mean precipitation rate, channel width, flood
              frequency, channel hydraulics. In case where the erodibility
              structure is turned on, this coefficient is applied to the reworked
              sediments. -->
        <erodibility>1.e-6</erodibility>>
        <!-- Number of steps used to distribute marine deposit.
             Default value is 5 (integer). -->
        <diffnb>10</diffnb>
        <!-- Proportion of marine sediment deposited on downstream nodes. It need
             to be set between ]0,1[. Default value is 0.9 (optional). -->
        <diffprop>0.05</diffprop>
        <!-- Critical density of water+sediment flux to trigger hyperpycnal curre
             off shore - (optional) -->
        <dens_cr>1060.</dens_cr>
        <!-- Deep basin depth under which hyperpycnal flow are forced to
             deposit [m] - (optional) -->
        <deepbasin>-2500.</deepbasin>
    </sp_law>
```

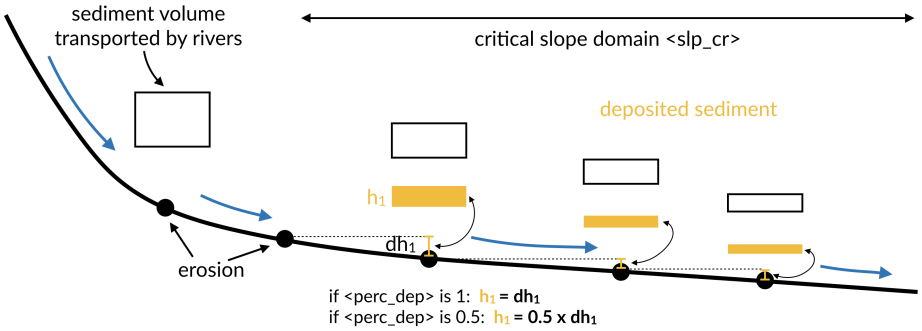
Depression – pit sedimentation

The first place where deposition will occur is in depression area. The code will first look for these regions and will estimate the volume of sediment required to fill the depression with a pit-filling algorithm (Planchon & Darboux, 2001). Then based on the available amount of sediment transported by the rivers to the depression and the value of the parameter <**fillmax**>, the filling will start. The parameter <**fillmax**> corresponds to the maximum elevation (in metres) of a potential lake that forms during one time step in **pyBadlands**. Decreasing the elevation of the lake will increase the number of iteration required to fill the depression, potentially increasing the resolution of the stratigraphic layers but in the same time increasing the model run time...



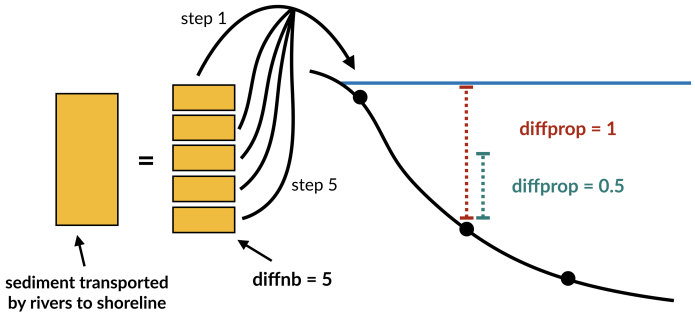
Alluvial plain forced deposition

The advantage of the detachment law, over the transport-limited approaches, is in the small restriction on computational time steps. The problem however, is that this detachment law is more suitable for mountainous regions and can not account for alluvial plain deposits. To simulate these deposits in **pyBadlands**, one can force the deposition when river beds reach a critical slope (<**slp_cr**>) and the amount of deposition is limited to a percentage of the maximum deposition that ensure no slope reversal (<**perc_dep**>).



Marine deposition

Once the sediment reaches the marine environment, rivers stop transporting them and sediment are usually deposited close to the shoreline. A diffusion law, defined within the hillslope structure, allows to diffuse marine sediments.
In addition to the diffusive transport, one can choose to transport the sediment further in the marine realm based on sediment delivery to coasts and marine slopes. To do so you will have to define two additional parameters (<**diffnb**> and <**diffprop**>).



<**diffnb**> is used to divide the initial volume in several equal parts that will be distributed iteratively over the model time step. <**diffprop**> relates to a proportion of the maximum thickness that can be deposited on a given nodes based on surrounding elevations.

In cases where hyperpycnal flows needs to be modelled the user needs to specify the critical density of water and sediment flux required to trigger these currents offshore. It enables to simulate submarine erosion and is set using the <**dens_crit**> parameter. To be enable <**deepbasin**> parameter will also need to be set and represent the depth under which hyperpycnal flows will be forced to deposit.

Transport-limited processes

```
<!-- Flux-dependent function structure  (optional)
     It is possible to modify the general detachment limited law to simulate c
     evolution governed by sediment flux-dependent bedrock incision rules.
     Visit tinyurl.com/badlands-incision for more information.
-->
<sedfluxfunction>
    <!-- Incision model type is defined with an integer between 0 and 4:
          + 0 - detachment limited (default) does not required to set additior
          + 1 - generalised undercapacity model (linear sedflux dependency) [c
          + 2 - parabolic sedflux dependency [tool & cover effect]
          + 3 - Turowski sedflux dependency [tool & cover effect]
          + 4 - saltation abrasion incision model
         See Hobley et al. (2011), JGR, 116 for more information.
    -->
    <modeltype>0</modeltype>
    <!-- Volumetric sediment transport capacity formulation is built with a st
         and requires the definition of 2 exponents for water discharge (mt) a
    <mt>1.5</mt>
    <nt>1.</nt>
    <!-- Transportability of channel sediment (erodibility coefficient) -->
    <kt>2.e-6</kt>
    <!-- Power law relation between channel width and discharge -->
    <kw>1</kw>
    <b>0.5</b>
    <!-- Erodibility dependence to the precipitation is defined with an expone
       Default value is set to 0. See Murphy et al. (2016), Nature, 532. -->
    <mp>0.</mp>
</sedfluxfunction>
```

In cases where the flow capacity to erode sediments needs to be taken into account, one needs to specify additional coefficients in the XmL input file. For a list of available transport-limited laws the user can refer to the chapter 1 of this manual on fluvial processes.

The volumetric sediment transport capacity (Q_t) is also defined using a power law function of unit stream power:

$$Q_t = \kappa_t (PA)^{m_t} S^{n_t} \tag{20}$$

where κ_t is a dimensional coefficient describing the transportability of channel sediment and m_t and n_t are dimensionless positive constants. A list of 4 different laws are available and summarised in the graph given in chapter 1. They are set using the <**modeltype**> element :

- 0 – detachment limited,
- 1 – undercapacity model (pure cover),
- 2 – tool & cover (almost parabolic),
- 3 – tool & cover (Turowski model), and
- 4 – saltation abrasion.

The only remaining parameter defined in the **sedfluxfunction structure** that can be used with the detachment limited model is the parameter <**mp**> corresponding to the coefficient **I** in the stream power law equation 1. All the other parameters correspond to the erodibility and coefficients values found in the transport-limited incision formulations.

It is worth noting that the values for the erodibilities vary quite substantially between the different laws (see table below) as well as for the exponent values.

Also in the case of transport-limited simulation, the maximum time step <**maxdt**> defined in the time structure needs to be reduced to avoid numerical instabilities.

Parameter		Value
m_t		1.5
n_t		1
K_t	m^{3-2mt}/yr	2.00×10^{-5}
K_{SP}	$m^{-(2m+1)}/yr$	4.00×10^{-5}
K_{SA}	$m^{-0.5}$	5.00×10^{-2}
K_{GA}	m^{-1}	7.00×10^{-3}
$m^{b,c,d}$		0.5, -0.25, 0
$n^{b,c,d}$		1, -0.5, 0
k_w	m^{1-3b}/yr^b	1
b		0.5

[a] transport-limited model
[b] detachment-limited stream power model
[c] saltation-abrasion model
[d] generalized abrasion model

Erodibility structure

The erodibility structure <**erocoeff**> allows to define a number of initial erodibility layers of varying spatial values. First one needs to define the number of layers to set for the simulation (<**erolayers**>).

```
<!-- Erodibility structure
     This option allows you to specify different erodibility values either on
     or within a number of initial stratigraphic layers. -->
<erocoeff>
    <!-- Number of erosion layers. -->
    <erolayers>4</erolayers>
    <!-- The layering is defined from top to bottom, with:
        - either a constant erodibility value for the entire layer or with an
        - either a constant thickness for the entire layer or with a thickness
    <!-- Constant erodibility and layer thickness -->
    <erolay>
        <!-- Uniform erodibility value for the considered layer. -->
        <erocst>3.e-6</erocst>
        <!-- Uniform thickness value for the considered layer [m]. -->
        <thcst>10</thcst>
    </erolay>
    <!-- Constant erodibility and variable layer thickness map -->
    <erolay>
        <!-- Uniform erodibility value for the considered layer. -->
        <erocst>3.e-6</erocst>
        <!-- Variable thicknesses for the considered layer [m]. -->
        <thmap>data/thlay2.csv</thmap>
    </erolay>
    <!-- Variable erodibilities and constant layer thickness -->
    <erolay>
        <!-- Variable erodibilities for the considered layer. -->
        <eromap>data/erolay3.csv</eromap>
        <!-- Uniform thickness value for the considered layer [m]. -->
        <thcst>30</thcst>
    </erolay>
    <!-- Variable erodibilities and thicknesses -->
    <erolay>
        <!-- Variable erodibilities for the considered layer. -->
        <eromap>data/erolay4.csv</eromap>
        <!-- Variable thicknesses for the considered layer [m]. -->
        <thmap>data/thlay4.csv</thmap>
    </erolay>
</erocoeff>
```

Each layer can be either of uniform erodibility values (<**erocst**>) or defined as a multi-erodibility map (<**eromap**>).

The definition also requires a thickness for each of these layers that can either be spatially constant (<**thcst**>) or variable (<**thmap**>).

Using a combination, one can produced a complex stack of spatially varying subsurface layers. The requirements for any of these maps are the same as for any other **pyBadlands** grids, i.e. ordered from the lower left corner to the upper right corner based on row-wise indexing (along the X-axis first).

Hillslope structure

```
<!-- Hillslope diffusion parameters:
     Parameterisation of the sediment transport includes the simple creep tra
     law which states that transport rate depends linearly on topographic gra
<creep>
    <!-- Surface diffusion coefficient [m2/a] -->
    <caerial>0.001</caerial>
    <!-- Marine diffusion coefficient [m2/a] -->
    <cmarine>0.005</cmarine>
    <!-- Critical slope for non-linear diffusion [m/m] - optional.
         Default value is set to 0 meaning non-lnear diffusion is not conside
    <cslp>0.8</cslp>
    <!-- River transported sediment diffusion
         coefficient in marine realm [m2/a] -->
    <criver>10.</criver>
    <!-- Critical slope above which slope failure are triggered [m/m] - optic
         Default value is set to 0 meaning non-lnear diffusion is not conside
    <sfail>0.26</sfail>
    <!-- Triggered failure sediment diffusion coefficient [m2/a] -->
    <cfail>3.</cfail>
</creep>
```

Transport along slope by gravity is simulated using 2 types of diffusion laws. In the first one, you can use a linear law commonly referred to as soil creep:

$$\frac{\partial z}{\partial t} = \kappa_{hl} \nabla^2 z \tag{21}$$

in which κ_{hl} is the diffusion coefficient and can be set for the marine (<**cmarine**>) and land (<**caerial**>) environments.

A second approach, based on a non-linear formulation, assumes that flux rates increase to infinity if slope values approach a critical slope S_c:

$$\frac{\partial z}{\partial t} = \nabla \cdot \frac{\kappa_{hn} \nabla z}{1 - (|\nabla z| / S_c)^2} \tag{22}$$

To use this approach, you will have to define another parameter (<**cslp**>) corresponding to the critical slope.

To increase marine transportation of freshly deposited river sediments along the coasts, one can decide to define an additional diffusion coefficient (<**criver**>) that will promote deep water transport of river-induced marine deposits.

Finally, one can choose to simulate slope failure or slump in aerial and marine environment by defining a critical slope value above which these processes are triggered (<**sfail**>) and a diffusion coefficient to transport the associated sediments (<**cfail**>).

Flexural isostasy structure

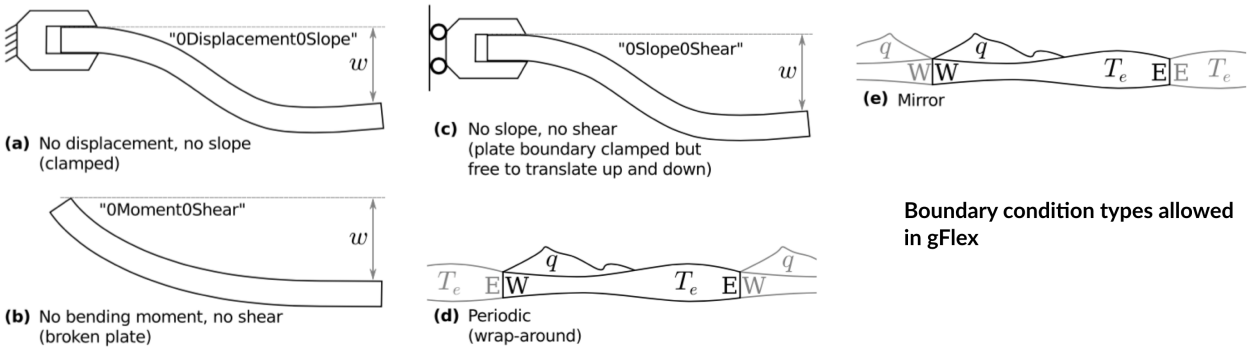
To estimate flexural isostasy, **gFlex** modular python package [64] is used in **pyBadlands**. It allows to compute isostatic deflections of Earth's lithosphere with uniform or non-uniform flexural rigidity and couple the interactions

with evolving surface loads induced by erosion/deposition associated to modelled surface processes.

The flexural isostasy is performed on a regular grid (defined based on the number of nodes along the X and Y axis – <**fnx**> and <**fny**>) and user-defined time intervals (<**ftime**>).

To compute the isostasy additional parameters are required: the mantle density (<**dmantle**>), the sediment density (<**dsediment**>), the Young's Modulus (<**youngMod**>), and the lithospheric elastic thickness.

```
<!-- Flexural isostasy parameters:
     Parameterisation of the flexural isostasy using the gFlex model from Wick
     The current wrapper limits the functionnality of the gFlex algorithm and
     the direct solver of the 2D finite difference method with the van Wees ar
     plate solution. -->
<flexure>
    <!-- Time step used to compute the isostatic flexure. -->
    <ftime>10000.0</ftime>
    <!-- Definition of the flexural grid:
         It is possible to setup a flexural grid at a resolution higher than t
         for the TIN to increase computational speed. In this case you need to
         discretization along X and Y axis. By default the same resolution as
         for the DEM file is used and the following 2 parameters are not requi
    <!-- Number of points along the X-axis - (optional)-->
    <fnx>100</fnx>
    <!-- Number of points along the Y-axis - (optional)-->
    <fny>100</fny>
    <!-- Mantle density [km/m3] -->
    <dmantle>3300</dmantle>
    <!-- Sediment density [km/m3] -->
    <dsediment>2500</dsediment>
    <!-- Young's Modulus [Pa] -->
    <youngMod>65E9</youngMod>
    <!-- The lithospheric elastic thickness (Te) can be expressed as a scalar
         a uniform thickness for the model area in this case the value is give
         parameter [m] - (optional) -->
    <elasticH>35000.</elasticH>
    <!-- In case where the lithospheric elastic thickness (Te) varies on the s
         you might want to use a grid defining for each points on the flexural
         Te value [m]. You will need to ensure that the grid dimensions match
         points given for the flexural grid resolution - (optional) -->
    <elasticGrid>data/elasticthickness.csv</elasticGrid>
    <!-- In case where the lithospheric elastic thickness (Te) varies with tim
         The elastic thickness relates to the age of the lithosphere with a si
         results from the square-root time dependence of lithospheric cooling
                      Te = a1 x sqrt(t) + a2
         You will need to define the coefficient for a1 and a2 where a1 is the
         and a2 the initial elastic thickness [m] at the start of the simulati
    <elasticA1>2.7</elasticA1>
    <elasticA2>10000.</elasticA2>
```



Boundary condition types allowed in gFlex

Three conditions can be set in regards to the elastic thickness. The simplest one assumes a uniform thickness over the simulated region (<**elasticH**>). The second defines a spatial variation in elastic thicknesses. This option requires an input file that needs to be of the same dimension as the flexural grid and needs to follow the conventional **pyBadlands** ordering approach (from the lower left corner to the upper right corner based on row-wise indexing). The last option considers a time dependent lithospheric elastic thickness T_e defined by the following equation:

$$T_e = A_1 e^t + A_2 \tag{23}$$

where t is the time in years, A_1 and A_2 the coefficients to define in the XmL file.

Finally **gflex** requires the definition of the boundary conditions along each borders (N,S,E,W) and 4 different types are available (as shown in the previous figure).
The proposed method consists in producing *snapshots* of wave-driven circulation distribution resulting from series of deep-water wave scenarios (<**climNb**>). These wave climates (<**climate**>) are based on a significant wave height (<**hs**>), a percentage of activity (<**perc**>) remaining fixed during the desired time interval (defined in years by <**start**> and <**end**> elements) and a mean wave direction (<**dir**>).

```
        <!-- Finite difference boundary conditions:
              + 0Displacement0Slope: 0-displacement-0-slope boundary condition
              + 0Moment0Shear: "Broken plate" boundary condition: second and
                        third derivatives of vertical displacement are 0. This
                        is like the end of a diving board.
              + 0Slope0Shear: First and third derivatives of vertical displacement
                        are zero. While this does not lend itsellf so easily to
                        physical meaning, it is helpful to aid in efforts to make
                        boundary condition effects disappear (i.e. to emulate the
                        NoOutsideLoads cases)
              + Mirror: Load and elastic thickness structures reflected at boundar
              + Periodic: "Wrap-around" boundary condition: must be applied to bot
                        North and South and/or both East and West. This causes, fo
                        example, the edge of the eastern and western limits of the
                        to act like they are next to each other in an infinite loo
             The boundary are defined for each edges W, E, S and N. -->
        <boundary_W>0Displacement0Slope</boundary_W>
        <boundary_E>0Displacement0Slope</boundary_E>
        <boundary_S>0Displacement0Slope</boundary_S>
        <boundary_N>0Displacement0Slope</boundary_N>
    </flexure>
```

Wave structure – beta testing

Wave evolution and associated sediment transport are calculated from a series of equations defined in Chapter 1. These equations are solved on a regular grid (resolution: <**wres**>) and at given time intervals (<**twave**>). Impact of wave on sediment transport is limited to shallow areas where water depth is below the wave base value (<**wbase**>).

The erosion thickness h_e (<**wEro**>) is limited to the top sedimentary layers and for simplicity is assumed to follow a logarithmic form:

$$h_e = C_e \ln(\tau_w / \tau_c) \text{ where } \tau_w > \tau_c \tag{24}$$

where C_e is an entrainment coefficient (<**wCe**>) controlling the relationship between shear stress and erosion rate.

Sediment is transported by wave-induced currents over a maximum number of iterations (<**tsetps**>). Remobilised sediment by wave are then diffused on the grid using a coefficient of diffusion <**wCd**> and during a number of iterations (<**dsteps**>).

```
<!-- Wave global parameters structure -->
<waveglobal>
    <!-- Wave interval [a] -->
    <twave>250.</twave>
    <!-- Wave grid resolution [m] -->
    <wres>1000.</wres>
    <!-- Maximum depth for wave influence [m] -->
    <wbase>20</wbase>
    <!-- Number of wave climate temporal events. -->
    <events>1</events>
    <!-- Mean grain size diameter [m] -->
    <d50>0.0001</d50>
    <!-- Wave sediment diffusion coefficient. Default is 50. -->
    <wCd>50.</wCd>
    <!-- Wave sediment entrainment coefficient. Value needs to be
         set between ]0,1]. Default is 0.5 -->
    <wCe>0.35</wCe>
    <!-- Maximum wave-induced erosion [m] -->
    <wEro>0.5</wEro>
    <!--  Steps used to perform sediment transport.
          Default is 1000. -->
    <tsteps>500</tsteps>
    <!--  Steps used to perform sediment diffusion.
          Default is 1000. -->
    <dsteps>500</dsteps>
</waveglobal>
```

```
 <!-- Wave definition based on wave global structure.
      The wave field needs to be ordered by increasing start time.
      The time needs to be continuous between each field without overlaps. -->
 <wave>
     <!-- Wave start time [a] -->
     <start>-14000.</start>
     <!-- Wave end time [a] -->
     <end>0</end>
     <!-- Wave climates number -->
     <climNb>3</climNb>
     <!-- Climatic wave definition for WaveSed model. -->
     <climate>
         <!-- Percentage of time this event is active during the time interval.
         <perc>0.3</perc>
         <!-- Significant wave height (in m) -->
         <hs>2.</hs>
         <!-- Wave direction in degrees (between 0 and 360) from the
              X-axis (horizontal) anti-clock wise. It specifies where the waves
              actually coming from. The wave directions are reduced to 8 possib
              East (dir = 0) - North (dir = 90) - West (dir = 180) - South (dir
              NE (0<dir<90) - NW (90<dir<180) - SW (180<dir<270) - SE (dir>270)
         <dir>0</dir>
     </climate>
     <!-- Climatic wave definition for WaveSed model. -->
     <climate>
         <!-- Percentage of time this event is active during the time interval.
         <perc>0.3</perc>
         <!-- Significant wave height (in m) -->
         <hs>2.</hs>
         <!-- Wave direction in degrees (between 0 and 360) from the
              X-axis (horizontal) anti-clock wise. It specifies where the waves
              actually coming from. The wave directions are reduced to 8 possib
              East (dir = 0) - North (dir = 90) - West (dir = 180) - South (dir
              NE (0<dir<90) - NW (90<dir<180) - SW (180<dir<270) - SE (dir>270)
         <dir>30</dir>
     </climate>
```

```
        <!-- Climatic wave definition for WaveSed model. -->
        <climate>
            <!-- Percentage of time this event is active during the time interval.
            <perc>0.4</perc>
            <!-- Significant wave height (in m) -->
            <hs>2.</hs>
            <!-- Wave direction in degrees (between 0 and 360) from the
                 X-axis (horizontal) anti-clock wise. It specifies where the waves
                 actually coming from. The wave directions are reduced to 8 possib
                 East (dir = 0) - North (dir = 90) - West (dir = 180) - South (dir
                 NE (0<dir<90) - NW (90<dir<180) - SW (180<dir<270) - SE (dir>270)
            <dir>300</dir>
        </climate>
    </wave>
```

Carbonate structure – beta testing

Carbonate platform formation in **pyBadlands** is still under testing and is provided without guarantee. Two different types of carbonates can be defined (<**species1**> & <**species2**>), in addition one can define hemipelagic deposition (<**pelagic**>). It is necessary to defined the region where carbonates will preferably grow using a basement map defining loose sediment (1) and hard cover (0). The map here again is a one column ASCII file ordered from the lower left corner to the upper right corner based on row-wise indexing.

```
    <!-- Specify species 1 growth functions based on 3 main controlling forces: de
         sedimentation rate and ocean wave height.
         These functions are defined as csv files produced using pre-processing IP
         notebook. -->

    <carb>
        <!-- Specify initial basement structure (0) for hard rock and (1) for loos
        <baseMap>data/base500south.csv</baseMap>
        <!-- Wave interval [a] -->
        <tcarb>50.</tcarb>
    </carb>

    <species1>
        <!-- Species 1 growth rate [m/yr]. -->
        <growth>0.009</growth>
        <!-- Depth control on species 1 evolution. -->
        <depthControl>data/depthcontrol1.csv</depthControl>
        <!-- Ocean wave height control on species 1 evolution. -->
        <waveControl>data/wavecontrolcarb1.csv</waveControl>
        <!-- Sedimentation control on species 1 evolution. -->
        <sedControl>data/sedcontrolcarb1.csv</sedControl>
    </species1>

    <!-- Specify species 2 growth functions based on 3 main controlling forces: de
         sedimentation rate and ocean wave height.
         These functions are defined as csv files produced using pre-processing IP
         notebook. -->
    <species2>
        <!-- Species 2 growth rate [m/yr]. -->
        <growth>0.005</growth>
        <!-- Depth control on species 2 evolution. -->
        <depthControl>data/depthcontrol2.csv</depthControl>
        <!-- Ocean wave height control on species 2 evolution. -->
        <waveControl>data/wavecontrolcarb2.csv</waveControl>
        <!-- Sedimentation control on species 2 evolution. -->
        <sedControl>data/sedcontrolcarb2.csv</sedControl>
    </species2>
```

```
    <!-- Specify pelagic deposition functions based on depth control.
         The function is defined as csv file produced using pre-processing IPythor
         notebook. -->
    <pelagic>
        <!-- Pelagic deposition rate [m/yr]. -->
        <growth>0.00005</growth>
        <!-- Depth control on pelagic deposition. -->
        <depthControl>data/pelagiccontrol.csv</depthControl>
    </pelagic>
```

Carbonate evolution is computed at user-defined interval (<**tcarb**>) and the carbonate production is controlled by a maximum of 3 parameters:

1. depth (i.e. accommodation – <**depthControl**>),
2. wave height (<**waveControl**>), and
3. sedimentation rate (<**sedControl**>).

These parameters are curves ranging between desired values of depth, sedimentation rate or wave height along the X-axis and [0,1] along the Y-axis. Each species is given a maximum vertical growth rate (<**growth**>) in metres per year which is then modulated based on the combination of each curve values.

Output structure

```
    <!-- Output folder path -->
    <outfolder>out</outfolder>
```

```
</badlands>
```

The <**outfolder**> element is optional but is highly recommended as it enables you to specify your ouput folder name. If not specified, the default name will be *output*. To prevent the deletion of any output folders if you have not changed the folder name, the code automatically creates a new name which add an underscore and a number at the end of the output filename. As an example, let us consider you have already ran a model with the <**outfolder**> element set to '*myexp*' and you have decided to change the erodibility value in the SPL law but kept the folder name the same. **pyBadlands** will create a new folder named '*myexp_0*'. If you keep changing any parameters omitting to change the folder name, you will have a list of folders like '*myexp_1*', '*myexp_2*', '*myexp_3*'...

Running models & visualisation

Quick start

Running on Mac OS X and Windows

You can run **pyBadlands** using Kitematic, a graphical user interface for running our Docker container. This can be downloaded and installed via the Docker Toolbox for both Mac OS X and Windows. You can have a look at this installation online guide for installing Docker.

Open Kitematic and create a new **pyBadlands** container by searching for **pyBadlands-demo**. Once the **pyBadlands** container has been loaded, launch the web preview to access the **pyBadlands/workspace** directory. You can then begin exploring the examples directories.

Running of Linux

pyBadlands is available through the Docker hub. For more information on Dockers and running on the command line please see the following wiki page.

Running from source code

For advanced users (developers, system administrators, etc) you can download the source code and build it. The **pyBadlands** source code is available from our github repository, please refer to the following Docker Files:

- dependencies: Dockerfile
- code: Dockerfile

for the latest details on dependencies and build-recipes. Guide for local installation are provided for OSX and Linux.

Using Kitematic to simplify Docker deployment

Docker allows us to distribute pre-built applications which are hosted in a virtual machine and are therefore platform independent. This simplifies things for us (only one platform we need to support) and for the users (you get a universal binary with complete reproducibility across platforms). However, if you are not running on a native Linux machine, you have to understand the way docker works before being able to access the binary.

Docker allows us to provide a single image to our users but not a consistent user experience. Kitematic is a Docker-supported user interface which allows users to discover and run Docker containers and interact with them in a predictable manner. With Kitematic you can download and run the **pyBadlands** containers through a gui which provides clickable links to the notebook server.

Getting started

Launching Kitematic for the first time brings up an app-store-like list of available containers. **pyBadlands** is available through the Docker hub and so can be discovered by searching for badlands and choosing **pybadlands-demo-serial** (Fig. 1).

Clicking *Create* on the **pybadlands-demo-serial** container will automatically download it and launch into the default configuration. This container is tested code from the release branch. If you want a specific version, check out the tags. This container is a rolling build which is updated from the development branch whenever there are changes. You can rename your container from the **General/Settings tab** which is not such a bad idea if you are using the latest build of a development branch because the container, once created, is an immutable snapshot of that version of the code.

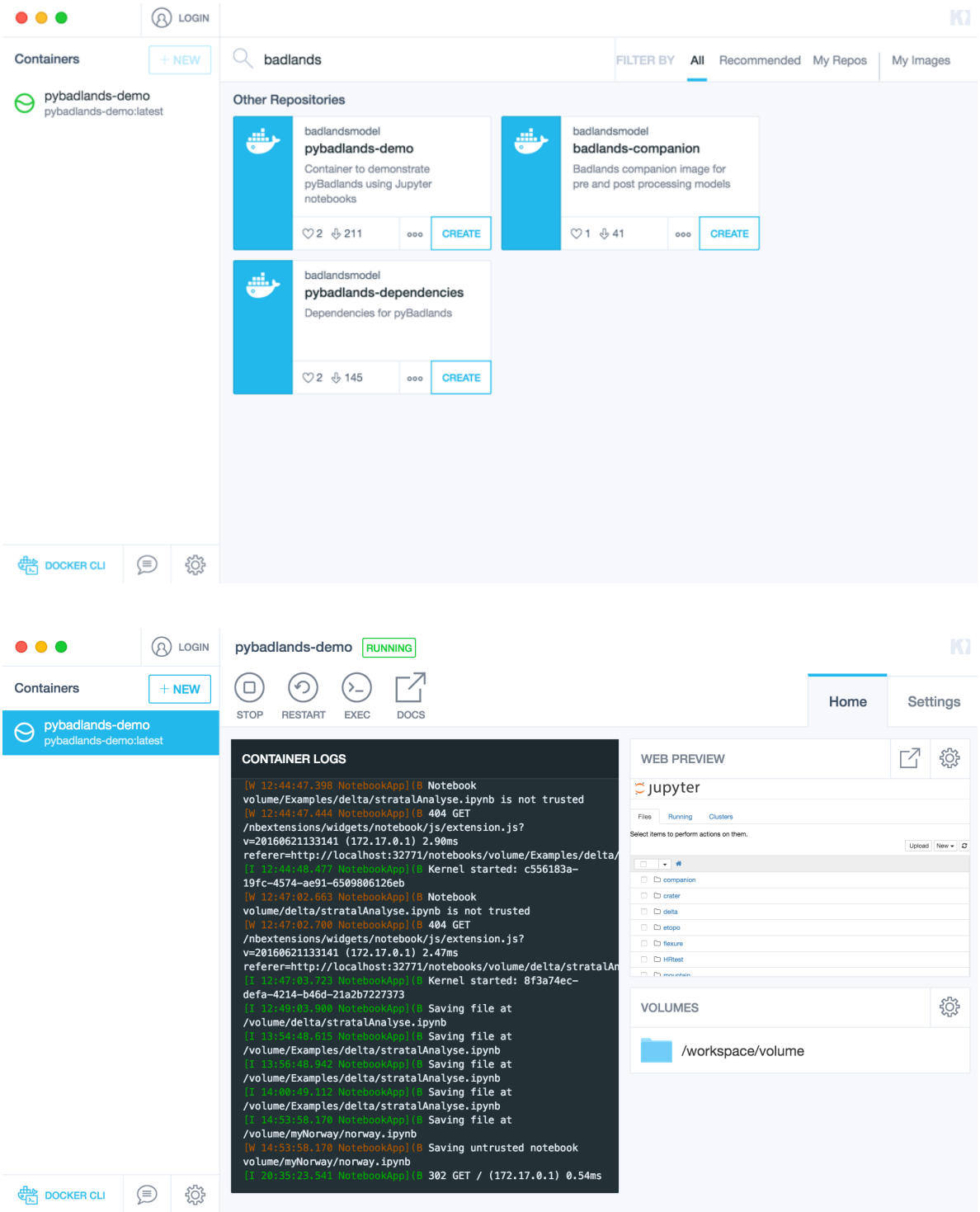


Fig 1.: Searching for **pyBadlands** container and running it.

Once **pyBadlands** is running, you will see a web-preview under the *Home* tab. In the windows alpha version this will be blank, but clicking on it will open the correct container in your browser.

The **pyBadlands** container is set to serve up the *Example folders* (notebooks) by default. If you click on the web-preview it will launch your default browser with the appropriate IP address and port (Fig 2 – you can change the port in the settings if you want).

How to work with your own notebooks using Kitematic

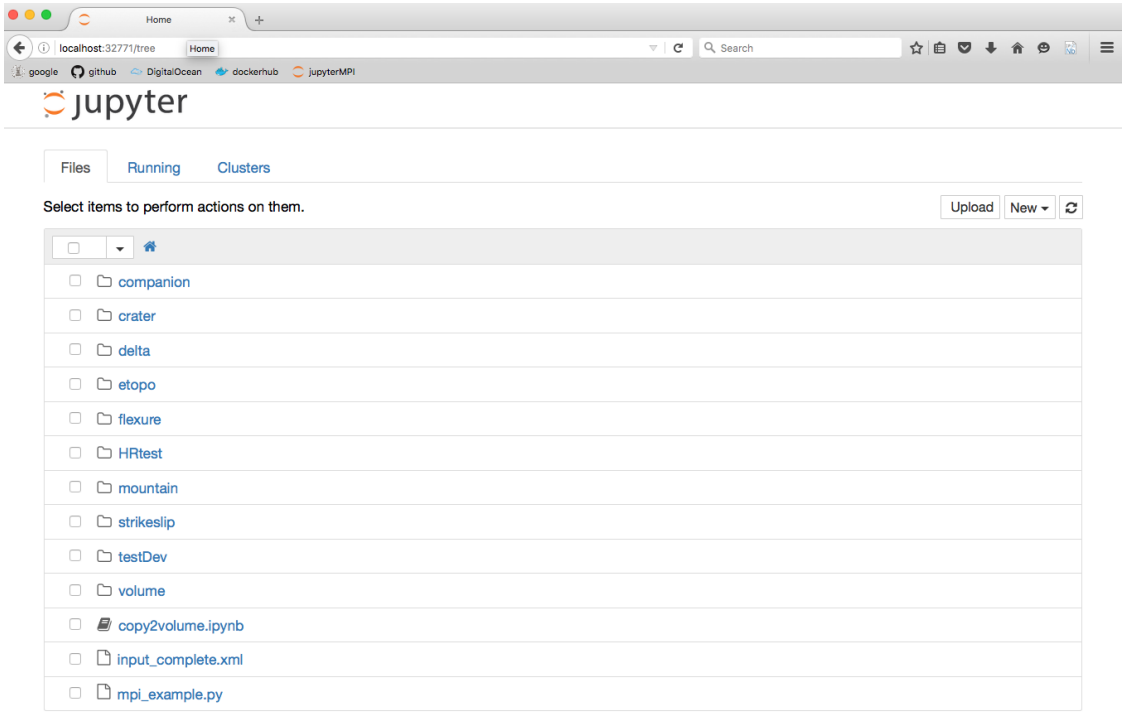


Fig 2.: Web-based environment for **pyBadlands**.

It does not make sense to keep all of your own work inside the virtual environment. If you want to manage your own data and keep notebooks *synchronised/versioned* in your own repository, and collaborate with others using shared space, then you can connect this space through to the virtual environment visible to the **pyBadlands** machine.

This is done through the settings tab of the container (Fig 3).

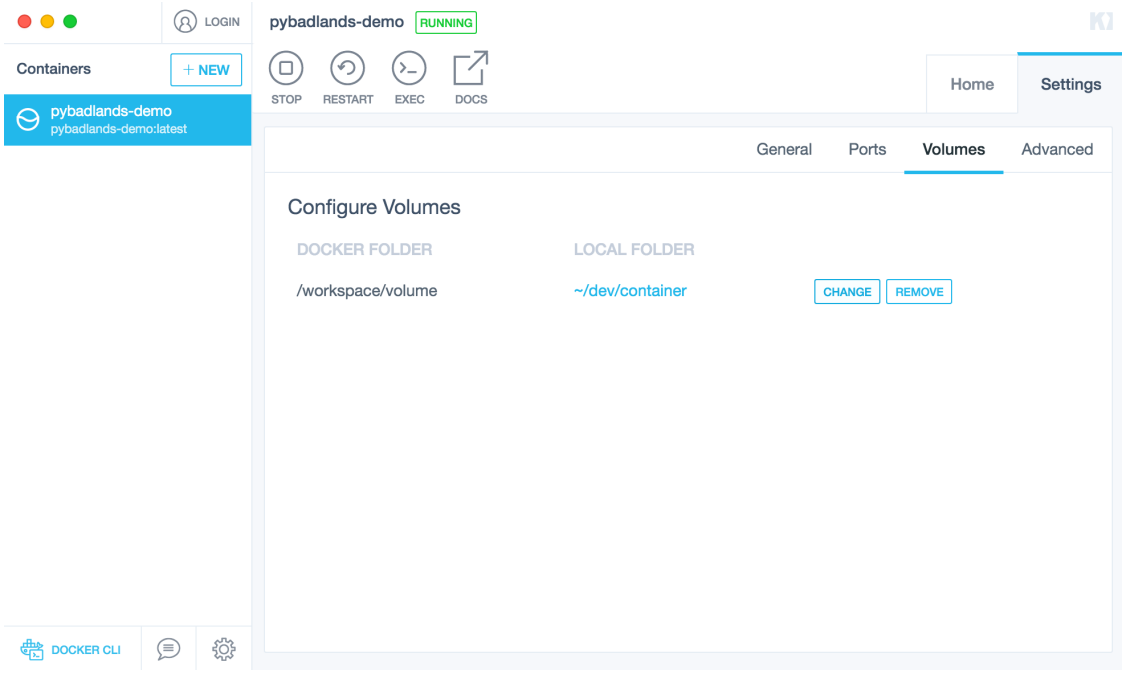


Fig 3.: Attaching a local volume to the container.

When the workspace is set to no folder, all data is written within the container itself. When you choose a folder in the outside world, this is the default workspace that the **pyBadlands** environment uses. Notebooks can be run from within that directory or its sub-directories and can also be accessed from the standard **filesystem**. If you have data which you want to share between different containers, you can point them all to some common space. The usual caveats apply though if you have several machines trying to use common files or write to a common directory.

You can access the contents of the **/workspace** volume to save any changes you make within the container because Kitematic tries to mount these locally in a default location see managing volumes for details. You will need to click the volume first to prompt the program to do this.

Jupyter environment

The **Jupyter Notebook** is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

For a complete guide on how to use Jupyter Notebooks it is recommended to look at the official on-line guide.

Here we will see some of the main functionalities relative to the use of **pyBadlands**.

pyBadlands Jupyter structure

Jupyter Notebook is officially supported for the following browsers:

- Chrome
- Safari
- Firefox

From experience, **Firefox** seems to give better integration that **Safari**.

When you launch **Jupyter notebook** through *Kitematic* the first page that you encounter is the **Notebook Dashboard**. This page is your **/home** directory and shows a file tree view of the folders and files installed on the container (as shown in Fig 4). The **volume** is the folder connected to your local architecture and the place where you will store your model input files and outputs to prevent any lost.

The *tree* is structured as follow:

- A series of Examples each defined in a specific folder (e.g. basin, crater, delta etc...),
- A input file (XmL) containing the entire list of parameters that can potentially be used,
- A *companion* folder listing all the notebooks that can be used for pre and post processing.

To visualise the XmL file in your web-browser, you will need to modify the **url** as follow:

http://localhost:32768/**view**/basin/basin.xml → http://localhost:32768/**edit**/basin/basin.xml

The complete list of experimental settings for each example is provided **here**, and for a look at some specific notebooks you can follow the links below:

- Mountain
- Delta
- eTopo
- Flexure
- Crater
- Strike-slip

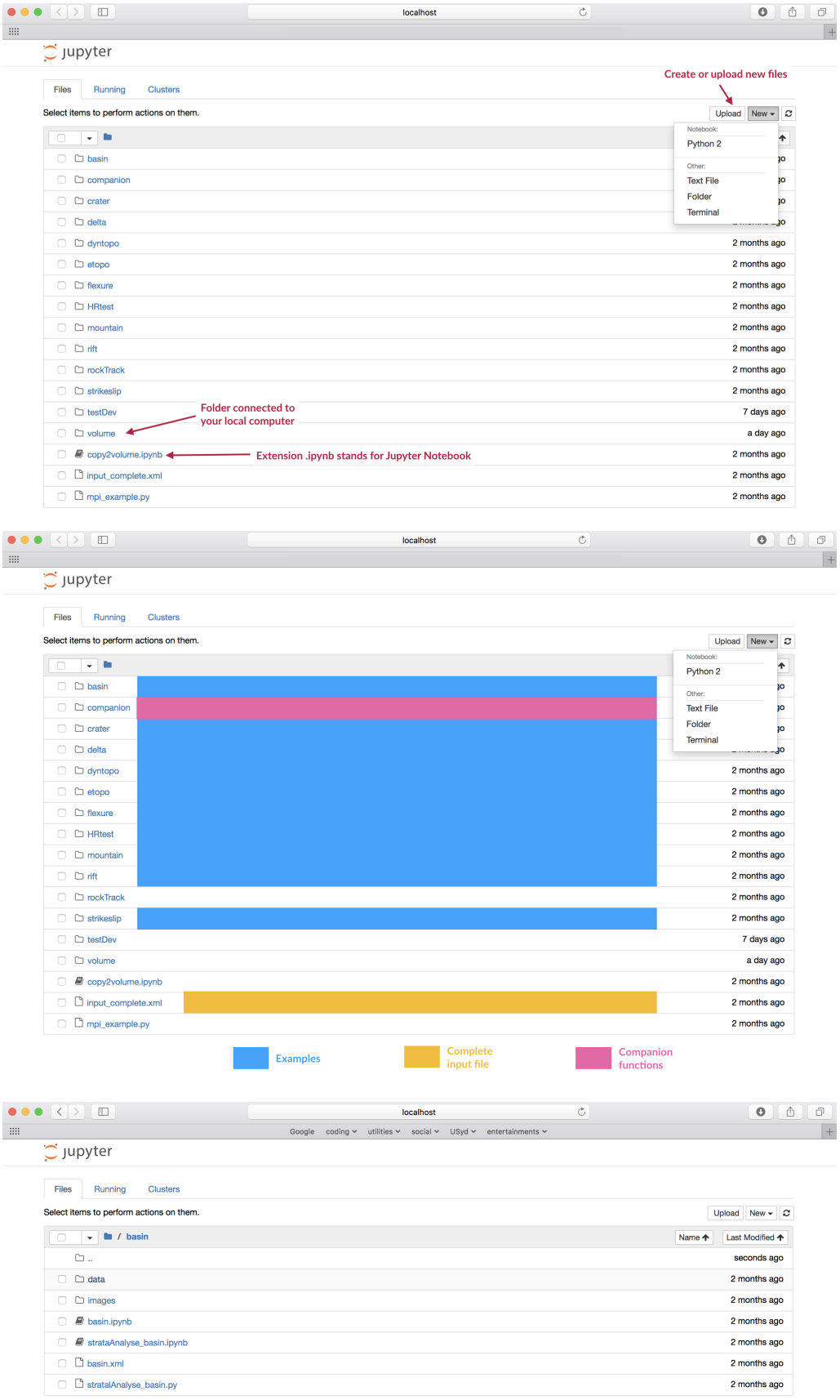


Fig 4.: Top - pyBadlands Notebook Dashboard, Middle – Tree structure – Example structure.

Each of the example folder follows a similar structure which is not required but which we recommend. It consists of a *data* folder containing your initial settings for example the topography grid, a sea-level curve, a tectonic file or a precipitation map. The input file (XmL) containing the functionalities that you want to turn-on in **pyBadlands**. And potentially additional *python scripts and notebook*s that will be used to define some initial conditions or analyse the model results.

Running a model

Once you have selected a Notebook file (*.ipynb* extension), the Notebook will open in what is called a *Notebook Editor*. This is where you will run your simulation.

IPython notebooks

The notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results. The IPython notebook combines two components:

- A web application: a browser-based tool for interactive authoring of documents which combine explanatory text, mathematics, computations and their rich media output.
- Notebook documents: a representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects.

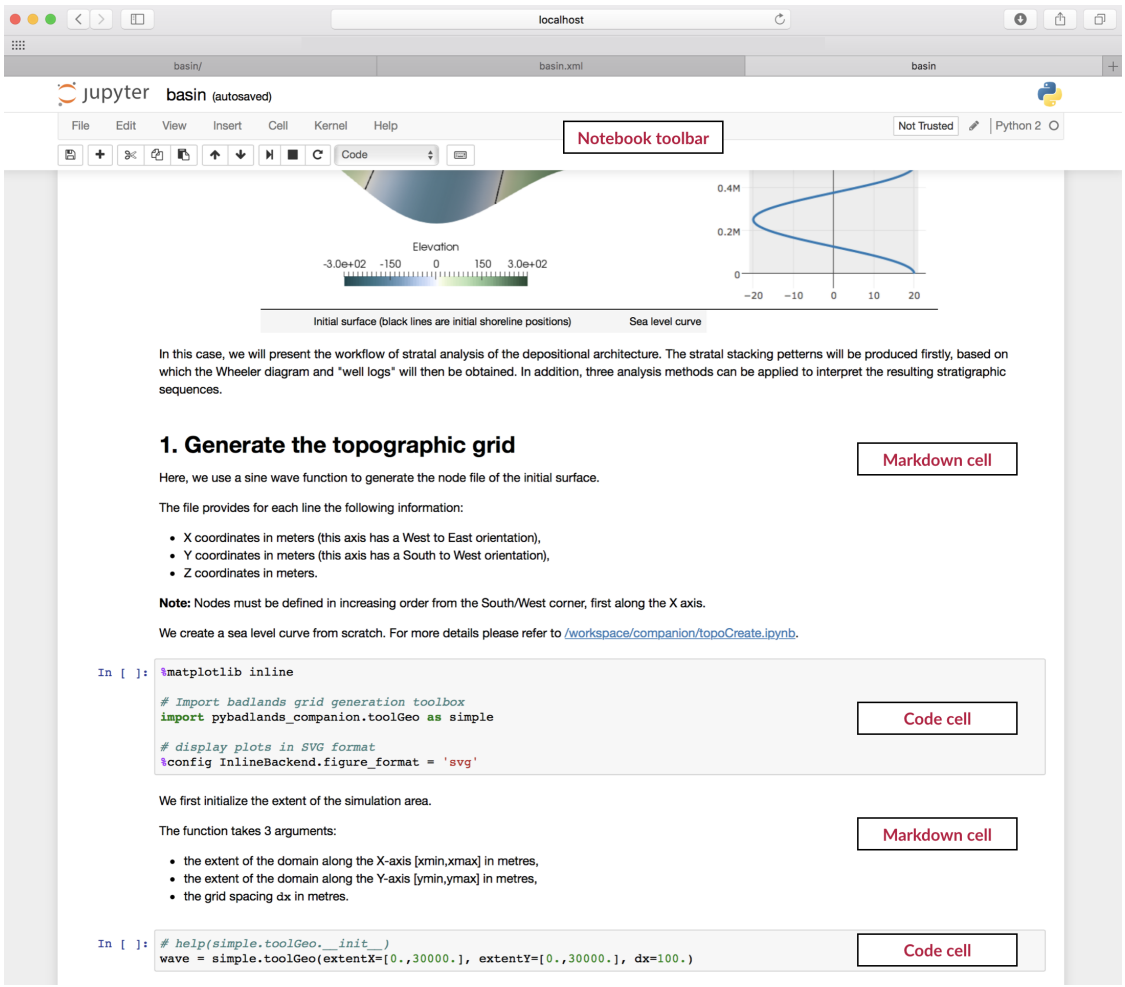


Fig 5.: Example of Notebook environment.

As shown in Fig 5, the *IPython notebooks* contains both documentation cells (a.k.a. *Markdown* cells) and code cells (in our case *Python2* commands). It allows you to define along with your model running command a series of documentation that will make your simulation easy to reproduce and to share with others. For an extensive user manual on IPython notebooks, you can visit the following link **Jupyter Notebook Users Manual**.

pyBadlands libraries

Running a model via the *IPython notebook* is simply done using the following set of commands:

```
In [*]: # import pyBadlands libraries
        from pyBadlands.model import Model as badlandsModel

        # initialise model
        model = badlandsModel()

        # load the XmL input file
        model.load_xml('input.xml')

        # run the model up to a given time [years]
        model.run_to_time(1000000)
```

```
    - Writing outputs (0.38 seconds; tNow = 0.0)
    - Writing outputs (0.23 seconds; tNow = 10000.0)
    - Writing outputs (0.24 seconds; tNow = 20000.0)
    - Writing outputs (0.23 seconds; tNow = 30000.0)
    - Writing outputs (0.25 seconds; tNow = 40000.0)
    - Writing outputs (0.24 seconds; tNow = 50000.0)
    - Writing outputs (0.26 seconds; tNow = 60000.0)
    - Writing outputs (0.25 seconds; tNow = 70000.0)
    - Writing outputs (0.26 seconds; tNow = 80000.0)
    - Writing outputs (0.27 seconds; tNow = 90000.0)
    - Writing outputs (0.27 seconds; tNow = 100000.0)
```

The first line is used to load **pyBadlands libraries** inside the notebook environment. The second line initialises a new model instance called **model** which contains pyBadlands' classes. For a complete list of functions and classes available within pyBadlands libraries the user can refer to the *readthedoc* online documentation. The third line loads the information in regards to your experiment in **pyBadlands** environment. This step parses the *XmL* file, initialises the model parameters and builds the initial surface. The last line is actually the one where the calculation is performed for a given number of time steps by defining a duration for your simulation. Once the model starts running it outputs some information in regards to the computation time and the number of years simulated.

Visualisation of model outputs

As the model starts to run it creates some simulation results which are located in the output folder name specified in the XmL file. The structure of this folder is shown in Fig 6.

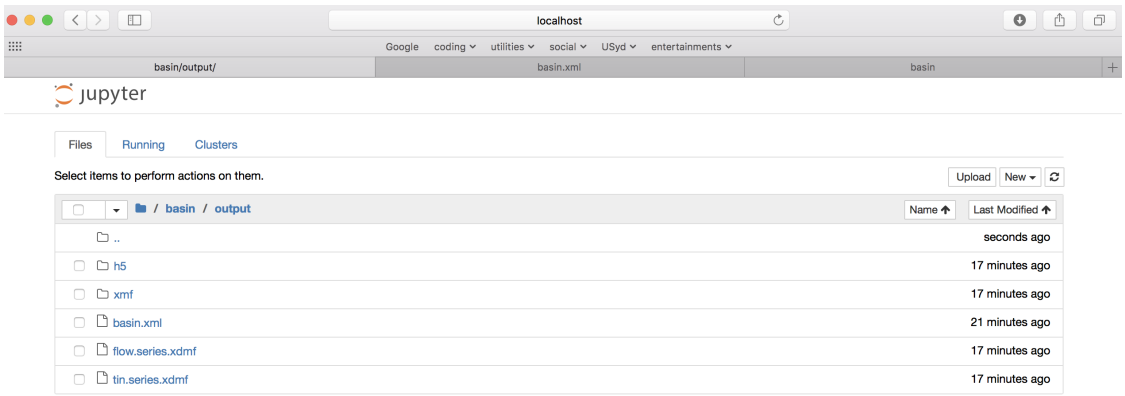


Fig 6.: Output folder from a **pyBadlands** simulation.

Model results consist of time series of surface evolution, river and catchment dynamics grids as well as underlying stratigraphic architecture mesh. These outputs are all produced as **Hdf5** binary files making it possible to

interact with multiple existing visualisation and analysis software, such as *Paraview* or directly via the *IPython notebook* interface. To read these **Hdf5** files, the code creates a *XML* schema (**.xmf** files) that tells how the data stored in the Hdf5 files need to be read. Then **XDMF** files format provides support for creation of an XML schema file.

Fig 6 shows the typical structure found in any pyBadlands output folder:

- **h5** folder contains the *HDF5* data, all the information computed by the model are stored in these files. You will have at least the tin (surface) and flow (stream network) dataset and also the sed (stratigraphy) data if the stratal structure is computed in your simulation.
- **xmf** folder contains the XML files used to read the H5 files contained in the h5 folder.
- the XML input file used to build this specific model.
- two **XDMF** files for the surface (*tin–series.xdmf*) and the flow network (*flow–series.xdmf*) which read the xmf files through time. These 2 files are the one you will use to look at your simulation results in Paraview.

Through an IPython notebook

Visualisation of the model results with IPython is a bit difficult due to the limited ability of the notebooks to handle 3D models in an interactive way. Nevertheless several post-processing functionalities do not require 3D plotting and here is two examples of IPython visualisation of some pyBadlands outputs. A list of available notebooks for visualisation of simulation outputs is available through the **Companion**.

Morphometric refers to quantitative description and analysis of the produced **pyBadlands** surface which could be applied to a particular kind of landform or to drainage basins and larger regions. The following suite of geomorphic attributes could be extracted:

- gradient: magnitude of maximum gradient,
- horizontal curvature describes convergent or divergent fluxes,
- vertical curvature: positive values describe convex profile curvature, negative values concave profile,
- aspect: direction of maximum gradient,
- discharge: it relates to the drainage area.

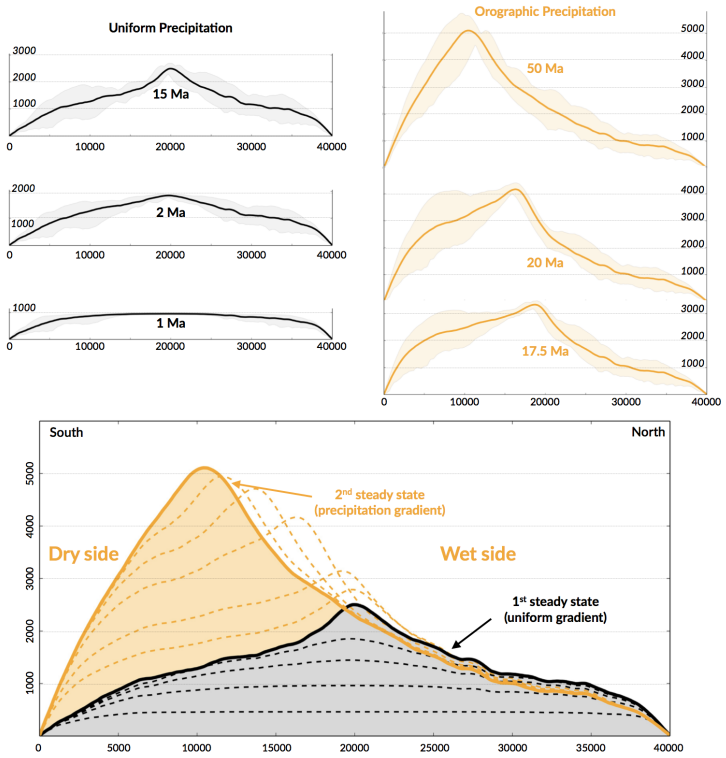


Fig 7.: Example of morphometric analyses with **pyBadlands** simulation.

Stratigraphic analyses When the stratigraphic structure is turned on in **pyBadlands** it is possible to:

- extract a cross-section,
- plot stratigraphic layers with different approaches,
- wheeler diagram,
- create for any given time step a 3D stratal mesh – VTK structured grid.

Using Paraview

Paraview is an open-source, multi-platform data analysis and visualization application. *ParaView* users can quickly build visualisations to analyse their data using qualitative and quantitative techniques.

To become an expert user in *Paraview* takes a lot of time and is beyond the scope of this documentation. The video below gives some of the main functionalities that are available to look at **pyBadlands** outputs in 3D.

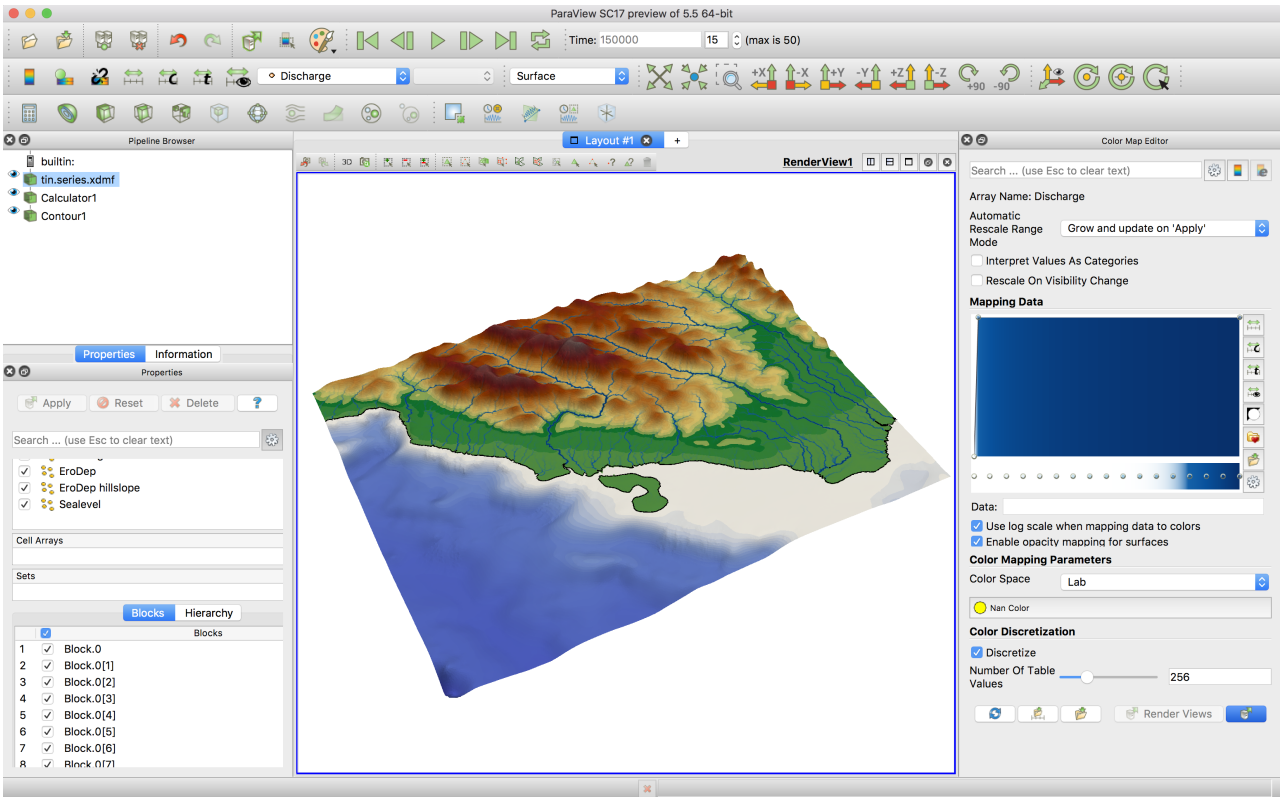


Fig 8.: Example of **Paraview** visualisation.

pyBadlands paraview quick start video

References

[1] K. X. Whipple and G. E. Tucker, "Implications of sediment-flux-dependent river incision models for landscape evolution," *Journal of Geophysical Research: Solid Earth*, vol. 107, no. B2, pp. 1–20, 2002.

[2] G. E. Tucker and G. R. Hancock, "Modelling landscape evolution," *Earth Surface Processes and Landforms*, vol. 35, no. 1, pp. 28–50, 2010.

[3] T. Salles and L. Hardiman, "Badlands: An open-source, flexible and parallel framework to study landscape dynamics," *Computers & Geosciences*, vol. 91, no. Supplement C, pp. 77–89, 2016.

[4] B. Campforts, W. Schwanghart, and G. Govers, "Accurate simulation of transient landscape evolution by eliminating numerical diffusion: the ttlem 1.0 model," *Earth Surface Dynamics*, vol. 5, no. 1, pp. 47–66, 2017.

[5] J. M. Adams, N. M. Gasparini, D. E. J. Hobley, G. E. Tucker, E. W. H. Hutton, S. S. Nudurupati, and E. Istanbulluoglu, "The landlab v1.0 overlandflow component: a python tool for computing shallow-water flow across watersheds," *Geoscientific Model Development*, vol. 10, no. 4, pp. 1645–1663, 2017.

[6] A. D. Howard, W. E. Dietrich, and M. A. Seidl, "Modeling fluvial erosion on regional to continental scales," *Journal of Geophysical Research: Solid Earth*, vol. 99, no. B7, pp. 13971–13986, 1994.

[7] D. E. J. Hobley, H. D. Sinclair, S. M. Mudd, and P. A. Cowie, "Field calibration of sediment flux dependent river incision," *Journal of Geophysical Research: Earth Surface*, vol. 116, no. F4, 2011.

[8] M. Attal, G. E. Tucker, A. C. Whittaker, P. A. Cowie, and G. P. Roberts, "Modeling fluvial incision and transient landscape evolution: Influence of dynamic channel adjustment," *Journal of Geophysical Research: Earth Surface*, vol. 113, no. F3, 2008.

[9] P. A. Cowie, A. C. Whittaker, M. Attal, G. P. Roberts, G. E. Tucker, and A. Ganas, "New constraints on sediment-flux dependent river incision: Implications for extracting tectonic signals from river profiles," *Geology*, vol. 36, pp. 535–538, 2008.

[10] L. S. Sklar and W. E. Dietrich, "Sediment and rock strength controls on river incision into bedrock," *Geology*, vol. 29, no. 12, pp. 1089–1090, 2001.

[11] J. M. Turowski, D. Lague, and N. Hovius, "Cover effect in bedrock abrasion: A new derivation and its implications for the modeling of bedrock channel morphology," *Journal of Geophysical Research: Earth Surface*, vol. 112, no. F4, 2007.

[12] T. Salles, N. Flament, and D. Müller, "Influence of mantle flow on the drainage of eastern australia since the jurassic period," *Geochemistry, Geophysics, Geosystems*, vol. 18, no. 1, pp. 280–305, 2017.

[13] G. E. Tucker and R. Slingerland, "Drainage basin responses to climate change.," *Water Resources Research*, vol. 33, no. 8, pp. 2031–2047, 1997.

[14] T. Salles, C. Griffiths, C. Dyt, and F. Li, "Australian shelf sediment transport responses to climate change-driven ocean perturbations.," *Marine Geology*, vol. 282, no. 3-4, pp. 268–274, 2011.

[15] D. E. J. Hobley, J. M. Adams, S. S. Nudurupati, E. W. H. Hutton, N. M. Gasparini, E. Istanbulluoglu, and G. E. Tucker, "Creative computing with landlab: an open-source toolkit for building, coupling, and exploring two-dimensional numerical models of earth-surface dynamics," *Earth Surface Dynamics*, vol. 5, no. 1, pp. 21–46, 2017.

[16] D. Granjeon and P. Joseph, *Concepts and applications of a 3D multiple lithology, diffusive model in stratigraphic modeling*. in: J. W. Harbaugh, W. L. Watney, E. C. Rankey, R. Slingerland, R. H. Goldstein & E. K. Franseen eds. Numerical Experiments in Stratigraphy: Recent Advances in Stratigraphic and Sedimentological Computer Simulations, vol. 62, SEPM Spec. Pub., Tulsa Ok, pp. 197–210., 1999.

[17] T. Salles and L. Hardiman, "Badlands: An open-source, flexible and parallel framework to study landscape dynamics.," *Comp. and Geosc.*, vol. 91, pp. 77–89, 2016.

[18] J. Braun and M. Sambridge, "Modelling landscape evolution on geological time scales: a new method based on irregular spatial discretization," *Basin Research*, vol. 9, no. 1, pp. 27–52, 1997.

[19] G. Tucker, S. Lancaster, N. Gasparini, and R. Bras, *The Channel-Hillslope Integrated Landscape Development Model (CHILD)*, pp. 349–388. Boston, MA: Harmon, Russell S. and Doe, William W. – Springer US, 2001.

[20] J. Braun and S. D. Willett, "A very efficient o(n), implicit and parallel method to solve the stream power equation governing fluvial incision and landscape evolution," *Geomorphology*, vol. 180–181, no. Supplement C, pp. 170–179, 2013.

[21] J. F. O'Callaghan and D. M. Mark, "The extraction of drainage networks from digital elevation data," *Computer Vision, Graphics, and Image Processing*, vol. 28, no. 3, pp. 323–344, 1984.

[22] A. Chen, J. Darbon, and J.-M. Morel, "Landscape evolution models: a review of their fundamental equations.," *Geomorphology*, vol. 219, p. 68–86, 2014.

[23] B. P. Murphy, J. P. L. Johnson, N. M. Gasparini, and L. S. Sklar, "Chemical weathering as a mechanism for the climatic control of bedrock river incision," *Nature*, vol. 532, p. 223, 2016.

[24] B. T. Crosby, K. X. Whipple, N. M. Gasparini, and C. W. Wobus, "Formation of fluvial hanging valleys: Theory and simulation," *Journal of Geophysical Research: Earth Surface*, vol. 112, no. F3, 2007.

[25] N. M. Gasparini, K. X. Whipple, and R. L. Bras, "Predictions of steady state and transient landscape morphology using sediment-flux-dependent river incision models," *Journal of Geophysical Research: Earth Surface*, vol. 112, no. F3, 2007.

[26] L. Sklar and W. E. Dietrich, *River Longitudinal Profiles and Bedrock Incision Models: Stream Power and the Influence of Sediment Supply*, pp. 237–260. American Geophysical Union, 1998.

[27] L. S. Sklar and W. E. Dietrich, "The role of sediment in controlling steady-state bedrock channel slope: Implications of the saltation-abrasion incision model," *Geomorphology*, vol. 82, no. 1, pp. 58–83, 2006.

[28] L. S. Sklar and W. E. Dietrich, "A mechanistic model for river incision into bedrock by saltating bed load," *Water Resources Research*, vol. 40, no. 6, 2004.

[29] G. Parker, *Somewhat less random notes on bedrock incision, Internal Memo. 118, St. Anthony Falls Lab.* Univ. of Minn.–Twin Cities, Minneapolis, 2004.

[30] R. A. DiBiase, K. X. Whipple, A. M. Heimsath, and W. B. Ouimet, "Landscape form and millennial erosion rates in the san gabriel mountains, ca," *Earth and Planetary Science Letters*, vol. 289, no. 1, pp. 134–144, 2010.

[31] I. J. Larsen and D. R. Montgomery, "Landslide erosion coupled to tectonics and river incision," *Nature Geoscience*, vol. 5, p. 468, 2012.

[32] G. Tucker and G. R. Hancock, "Modelling landscape evolution.," *Earth Surf. Process Landf.*, vol. 35, no. 1, p. 28–50, 2010.

[33] T. Salles and G. Duclaux, "Combined hillslope diffusion and sediment transport simulation applied to landscape dynamics modelling.," *Earth Surf. Process Landf.*, vol. 40, no. 6, p. 823–39, 2015.

[34] G. E. Tucker and D. N. Bradley, "Trouble with diffusion: Reassessing hillslope erosion laws with a particle-based model," *Journal of Geophysical Research: Earth Surface*, vol. 115, no. F1, 2010.

[35] E. Foufoula-Georgiou, V. Ganti, and W. E. Dietrich, "A nonlocal theory of sediment transport on hillslopes," *Journal of Geophysical Research: Earth Surface*, vol. 115, no. F2, 2010.

[36] D. J. Andrews and R. C. Bucknam, "Fitting degradation of shoreline scarps by a nonlinear diffusion model," *Journal of Geophysical Research: Solid Earth*, vol. 92, no. B12, pp. 12857–12867, 1987.

[37] J. J. Roering, J. W. Kirchner, L. S. Sklar, and W. E. Dietrich, "Hillslope evolution by nonlinear creep and landsliding: An experimental study," *Geology*, vol. 29, no. 2, pp. 143—-146, 1999.

[38] J. J. Roering, J. W. Kirchner, and W. E. Dietrich, "Hillslope evolution by nonlinear, slope-dependent transport: Steady state morphology and equilibrium adjustment timescales," *Journal of Geophysical Research: Solid Earth*, vol. 106, no. B8, pp. 16499–16513, 2001.

[39] G. B. Airy, *On tides and waves.* Encyclopaedia Metropolitana, 5, 241., 1845.

[40] G. G. Stokes, *On the theory of oscillatory waves.* Transactions of the Cambridge Philosophical Society, 8, p. 441., 1847.

[41] D. M. Tetzlaff, "Modelling Coastal Sedimentation through Geologic Time.," *Journal of Coastal Research*, vol. 21, no. 3, p. 610–617, 2005.

[42] R. L. Soulsby, "Bed shear stress due to combined waves and currents.," tech. rep., Advances in Coastal Morphodynamics, edited by M. J. F. Stive et al., pp. 4:20-4:23, Delft Hydraulics, Delft, The Netherlands, 1995.

[43] M. O. Green and G. Coco, "Review of wave-driven sediment resuspension and transport in estuaries.," *Rev. Geophys.*, vol. 52, p. 77–117, 2014.

[44] R. L. Soulsby, L. Hamm, G. Klopmann, D. Myrhaug, R. R. Simons, and G. P. Thomas, "Wave-current interaction within and outside the bottom boundary layer.," *Coastal Eng.*, vol. 21, p. 41–69, 1993.

[45] M. S. Longuet-Higgins, "Longshore currents generated by obliquely incident sea waves.," *J. Geophys. Res.*, vol. 75, no. 33, pp. 1–35, 1970.

[46] R. L. Soulsby, *Dynamics of Marine Sands.* pp. 429, Thomas Telford, London, U.K., 1997.

[47] P. D. Komar and M. C. Miller, "The initiation of oscillatory ripple marks and the development of plane-bed at high shear stresses under waves.," *J. Sed. Res.*, vol. 45, no. 3, pp. 697–703, 1975.

[48] L. C. Van Rijn, "Sediment Transport, Part I: Bed Load Transport.," *Journal of Hydraulic Engineering*, vol. 110, no. 10, pp. 1431–1456, 1984.

[49] G. M. D. Warrlich, D. Waltham, and D. Bosence, "Quantifying the sequence stratigraphy and drowning mechanisms of atolls using a new 3-D forward modelling program (CARBONATE 3D).," *Basin Research*, vol. 14, pp. 379–400, 2002.

[50] G. M. D. Warrlich, D. Bosence, D. Waltham, C. Wood, A. Boylan, and B. Badenas, "3D stratigraphic forward modelling for analysis and prediction of carbonate platform stratigraphies in exploration and production.," *Marine and Petroleum Geology*, vol. 25, pp. 35–58, 2008.

[51] S. J. Barrett and J. M. Webster, "Reef Sedimentary Accretion Model (ReefSAM): Understanding coral reef evolution on Holocene time scales using 3D stratigraphic forward modelling.," *Marine Geology*, vol. 391, pp. 108–126, 2017.

[52] R. V. Demicco, "CYCOPATH 2-D, a two-dimensional, forward-model of cyclic sedimentation on carbonate platforms.," *Comput. Geosci.*, vol. 24, pp. 405–423, 1998.

[53] R. V. Demicco and G. J. Klir, "Stratigraphic simulations using fuzzy logic to model sediment dispersal.," *J. Petroleum Science and Engineering*, vol. 31, pp. 135–155, 2001.

[54] T. Hattab, F. Ben Rais Lasram, C. Albouy, C. Sammari, M. S. Romdhane, P. Cury, F. Leprieur, and F. Le Loc'h, "The Use of a Predictive Habitat Model and a Fuzzy Logic Approach for Marine Management and Planning.," *PLoS ONE*, vol. 8, no. 10, 2013.

[55] A. Collin, K. Nadaoka, and L. Bernardo, "Mapping the Socio-Economic and Ecological Resilience of Japanese Coral Reefscapes across a Decade.," *ISPRS Int. J. Geo-Inf.*, vol. 4, no. 3, pp. 900–927, 2015.

[56] U. Nordlund, "FUZZIM: Forward stratigraphic modeling made simple.," *Comp. and Geosc.*, vol. 25, pp. 449–456, 1999.

[57] E. H. Meesters, R. P. M. Bak, S. Westmacott, M. Ridgley, and S. Dollar, "A fuzzy logic model to predict coral reef development under nutrient stress," *Conserv. Biol.*, vol. 12, pp. 957–965, 1998.

[58] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[59] C. Thieulot, P. Steer, and R. S. Huismans, "Three-dimensional numerical simulations of crustal systems undergoing orogeny and subjected to surface processes," *Geochemistry, Geophysics, Geosystems*, vol. 15, no. 12, pp. 4936–4957, 2014.

[60] B. U. Haq, J. Hardenbol, and P. R. Vail, "Chronology of fluctuating sea levels since the Triassic (250 million years ago to present).," *Science*, vol. 235, pp. 1156–1167, 1987.

[61] K. G. Miller, M. A. Kominz, J. V. Browning, J. D. Wright, G. S. Mountain, M. E. Katz, P. J. Sugarman, B. S. Cramer, N. Christie-Blick, and S. F. Pekar, "The phanerozoic record of global sea-level change.," *Science*, vol. 310, pp. 1293–1298, 2005.

[62] R. B. Smith and I. Barstad, "A linear theory of orographic precipitation," *Journal of the Atmospheric Sciences*, vol. 61, no. 12, pp. 1377–1391, 2004.

[63] J. R. Shewchuk, *Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator*, pp. 203–222. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996.

[64] A. D. Wickert, "Open-source modular solutions for flexural isostasy: gflex v1.0," *Geoscientific Model Development*, vol. 9, no. 3, pp. 997–1017, 2016.

[65] W. Helland-Hansen and G. Hampson, "Trajectory analysis: concepts and applications," *Basin Research*, vol. 21, no. 5, pp. 454–483, 2009.

[66] J. Neal and V. Abreu, "Sequence stratigraphy hierarchy and the accommodation succession method," *Geology*, vol. 37, no. 9, pp. 779–782, 2009.

[67] J. E. Neal, V. Abreu, K. M. Bohacs, H. R. Feldman, and K. H. Pederson, "Accommodation succession (δa/δs) sequence stratigraphy: observational method, utility and insights into sequence boundary formation," *Journal of the Geological Society*, vol. 173, no. 5, pp. 803–816, 2016.