Hillslope 1D : Python

Boussinesq's model initially coded by Jean Marçais on Matlab was ported to Python (3.5)

I- Model Principle:

The code is based on Boussinesq's equation. It is taken from Jean Marçais' work (done on matlab).

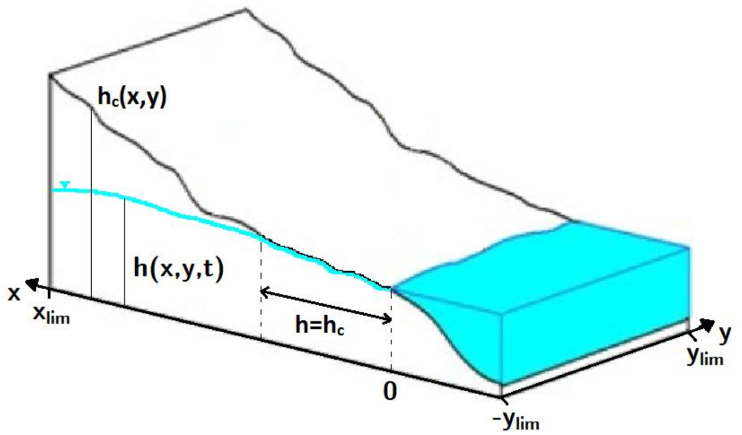

Figure 1 Schematic representation of the hillslope used in the model

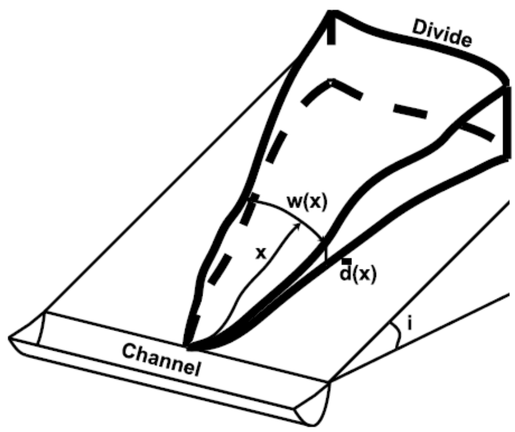

Figure 2 Spatial structure of the hillslope

Fig.1 and Fig.2 present hillslope’s structure in the model based on Marçais’ work.

II- Classes :

II-1- Classes hierarchy :

A master class (BoussinesqSimulation) carry the other classes as attributes. Attributes and methods of all classes are listed after.

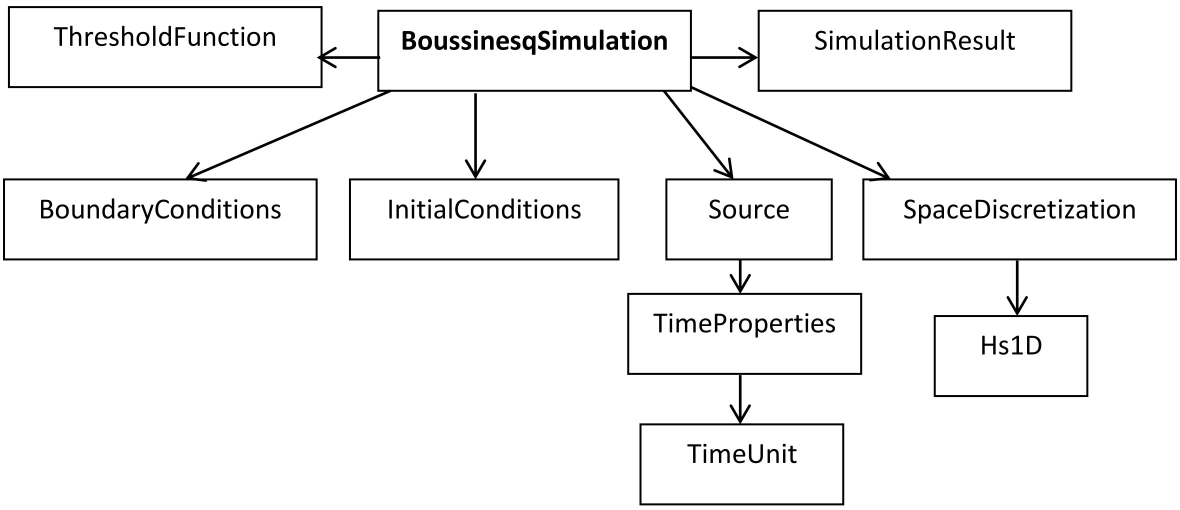

Figure 3 Hierarchy of used classes

II-2- Classes attributes :

List of all classes attributes and their meaning.

- **BoussinesqSimulation :**

  - Id : An identifier of the modeled hillslope
  - model : Implicit Problem to be solved
  - sim : simulation of the Implicit Problem using IDA integrator
  - k : the hydraulic conductivity of the modeled hillslope
  - f : the kinematic porosity of the hillslope
  - m : the mass matrix of the DAE

- **BoundaryConditions** :

  - boundary_type : list containing the two boundaries of the system (upstream and downstream) : either known stock or imposed flow.
  - boundary_value : values corresponding to the two boundary conditions
  - edges : matrix used in calculation to describe boundary types and values
  - edges_bool : matrix used in calculation describing only the type of boundary

- **InitialConditions** :

  - percentage_loaded : describes the percentage of stock initially stored in cells
  - qin : initial flow between all cells (defined over the edges)
  - sin : initial stock in each cell (defined over the nodes)
  - q_sin : initial seepage (overland flow) (defined over the nodes
  - Smax : maximal stock that can be stored in each cell (defined over the nodes) (depends on the soil_depth and the porosity)

- **Source** :

  - tmax : maximal time value of the series
  - recharge_type : type of the recharge used in the model (custom, periodical, squared)
  - recharge_chronicle : recharge time serie used
  - period : period of the recharge time serie (if no custom)
  - recharge_rate : recharge flux in m/s
  - t : time serie describing the system

- **Time Properties** :

  - Nt : number of time steps
  - tmin : minimal time value
  - tmax : maximal time value
  - unit : time unit of tmax and tmin
  - t : time serie

- **TimeUnit** :

  - tmax : maximal time value
  - unit : time unit

- **SpaceDiscretization** :

  - N_nodes : number of cells nodes used to define the hillslope
  - N_edges : number of cells edges used to define the hillslope (N_nodes + 1)
  - x_node : coordinates of cells nodes
  - x_edges : coordinates of cells edges
  - dx_node : distance between two consecutive cell's nodes
  - dx_edges : distance between two consecutives cell's edges
  - angle_node : slope of the hillslope (defined over nodes)
  - soil_depth_node : depth of each cell (defined over nodes)
  - w_node : width of the cells (defined over nodes)
  - xmax : maximal coordinate (edge)
  - xmin : minimal coordinate (edge)
  - a : matrix for conversion from nodes to edges
  - b : matrix for conversion from edges to nodes

- omega : weight matrix
- omega 2 : other weight matrix
- xcustom : list of customed coordinates (-1 if not active)
- dicretzation : discretization type of the hillslope (lienar, logarithmic, square)

➢ **Hs1D** :

- soil_depth_edges : thickness of each cell (defined over edges)
- w_edges : width of each cell (defined over edges)
- angle_edges : slope of the hillslope for each cell (defined over edges)
- k : hydraulic conductivity
- f : kinematic porosity

➢ **SimulationResults** :

- S : Stock in each cell of the hillslope for each time step (over nodes)
- Q : Flow in each cell of the hillslope for each time step (over edges)
- QS : Seepage in each cell of the hillslope for each time step (over nodes)
- x_node : coordinates of cells nodes
- x_edges : coordinates of celles edges
- t : list of time steps

II-2- Classes main methods :

List of main classes methods and their functions.

➢ **BoussinesqSimulation :**

- set_initial_conditions(percentage_loaded, w, soil_depth, f) : defines initial conditions of stock and flow over the hillslope based on percentage_loaded, boundary conditions and geometry of the system
- compute_q_from_s : compute flow rate (edges) based on darcy's law and stock (nodes)
- compute qs_from_q : compute seepage from flow rate
- compute_dsdt_from_q : compute stock variation from flow rate
- compute_c : compute the matrix C used in the DAE as a multiplier of y. Computed from S, Q and QS
- test_derivative : Test to determine if stock is still positive and seepage is occuring or not
- compute_alpha : compute a matrix defining variations over time. Used to compute Q, S and QS
- compute_beta : compute a matrix defining variations over time. Used to compute Q, S and QS

- compute_source_terms : compute the recharge for a time step on each cell based on recharge defined by user
- rhs : Compute differential equation dy/dt = C*dy + d
- res : Compute algebraic differential equation m * dy/dt = C*dy + d
- implicit_scheme_solver : Resolution of the DAE using implicit problem solver DAE
- compute_mass_matrix : Compute the matrix m of the DAE
- output_simu : Write Simulations Results (Q,S,QS and x_Q,x_S,t_res) in .txt files (delimiter : tab) in the current working directory

- **BoundaryConditions :**

- fixed_edges_matrix_boolean : creates a four terms matrix which contains either 0 or 1, depending on the type of boundary for the two limits of the system.
- fixed_edge_matrix_values : does exactly the same as fixed_edge_matrix_boolean, except that the values correspond to the value of the boundary conditions for each limit.

- **InitialConditions :**

No methods except the constructor which contains all assignments.

- **Source**

- source_terms : defines the recharge of the system based on recharge_type in order to assign a period.
- set_recharge_chronicle : computes the recharge time serie based on the recharge type, value and period (for all time steps)
- compute_recharge_rate : used to compute the recharge's value on each time step.

- **TimeProperties :**

- time_properties : creates a regularly spaced vector containing each time step location based on tmin, tmax and Nt

- **TimeUnit :**

- time_to_seconds : convert time to seconds from other units, based on unit and tmax.
- time_to_days : same as time_to_seconds but to days
- time_to_years : same as time_to_seconds but to years
- time_to_hours : same as time_to_seconds but to hours

- **SpaceDiscretization :**

  - space_discretiization : computes coordinates of edges based on user's choices, xcustom, xmin, xmax, N_edges.
  - resample_hs1D_spatial_variables : computes w, angle, soil_depth and x for nodes based on values for edges and interpolation.
  - get_angle_node : computes angle v&alues on nodes using resample_hs1D_spatial_variables
  - get_w_node : same as get_angle_node but for w
  - get_soil_depth_node : same as get_angle_node but for soil_depth
  - set_matrix_properties : computes a, b, omega and omega2 using the corresponding methods
  - compute_x_node : compute x_node based on x_edges as the center of two consecutive edges
  - compute_dx_edges : computes the distance between two consecutive edges
  - compute_dx_node : compute the distance between two consecutive nodes
  - first_derivative_upstream : compute b, the conversion matrix from edges to nodes
  - first_derivative_downstream : computes a, the conversion matrix from nodes to edges
  - first_derivative_centered : computes another conversion matrix (UNUSED)
  - weight_matrix : computes omega
  - weight_matrix_bis : computes omega2 (UNUSED)

- **Hs1D :**
  - get_w_edges : returns w_edges
  - get_soil_depth_edges : returns soil_depth_edges
  - get_angle_edges : returns angle_edges
  - get_k : return k
  - get_f : returns f

- **SimulationResults :**

  No methods except the constructor which regroups all attributes assignment.

III – Calculation sequence

III-1- Initialization

First the model is initialized using inputs:

- Building of the spatial structure of the hillslope : x_edges, x_node, w_edges, w_node, soil_depth_edges, soil_depth_node, angle_edges, angle_node using **SpaceDiscretization** and **Hs1D**.
- Setting boundary conditions using **BoundaryConditions**
- Setting initial values of S, Q and QS (sin, qin, q_sin) using **InitialConditions** based on percentage_loaded, boundary conditions and spatial structure.
- Setting time properties and source terms of the hillslope using **TimeProperties** and **Source**.

III-2- Building the model

Model to solve is built using implicit_scheme_solver from **BoussinesqSimulation**. It's built using ImplicitProblem class from assimulo library.

III-3- Solving the DAE

DAE is solved using implicit_scheme_solver from **BoussinesqSimulation** using IDA from assimulo

III-4- Output of integration

First, results of integration are stored using **SimulationResults**. Then, text files (separator \t) are created containing spatial and temporal properties of the hillslope and integration results using output_simu from **BoussinesqSimulation**.

A file named "**test_func.py**" is used to test the resolution on a typical slope.