Hydrodynamic model in GAMA

Last Update: 07 Mars 2013, by Benoit Gaudou

Model version: modelv3

GAMA version: GAMA 1.6 (tested on release 6061 from the SVN)

Contenu

Lexics

In the sequel, we use following notations:

- H : the water level (the height from the sea level) (m)
- A : the area of water in a section (m^2)
- Q : the water flow rate (m^3.s^-1)
- R : the water height from the bottom of the section (m)
- s : the slope of channel bed (this represents the slope from a given section to the previous one)
- n : roughness factor (we take: n = 0.02)
- H0 : the level height at the downstream
- Q0 : the water flow rate at the upstream
- Downstream : the lowest section of the considered part of river
- Upstream : the highest point in the considered part of river

Data Used

For the model, we use data produced by ISIS software. The model uses three distinct data sources:

- a shapefile located at "../includes/mythanh_river.shp" that contains data about the river. The attribute table contains:
    - ID : the identification number of each river polyline
- a shapefile located at "../includes/section_mt.shp" that contains data about sections. Note that the first and the last sections are repeated twice each with different NODE_TYPE. The attribute table contains attributes:
    - NODE_LABEL : the name of the section,
    - NODE_TYPE : the type of the node among QTBDY, HTDBY and RIVER_SECTION
    - EASTING
    - NORTHING
- A sqllite database located at "../includes/hydro.db". The sqllite database includes 2 tables:
    - **sections**, with attributes
        - NAME
        - X
        - Y
    - **water_levels**, with attributes
        - Water_level
        - Time_step

Note that the shapefile **NODE_LABEL** attribute corresponds to the **NAME** attribute of the sections table.

Description of the model: a steady model

Hypothesis:

- Steady case (Q is constant)
- Data hypothesis: sections in shapefile are ordered.

- Implementation hypothesis: section3D are ordered from upstream to downstream and used in this order
- Computation of the R makes the hypothesis that the section shape is closed to a rectangle.

Aim

- Compute the H for each section from the downstream to the upstream

Input of the model

- A shapefile to locate the section on the river
- Sections list of points (read from a sqlite database file)
- H0 and Q0 time series

Description of agents

Various agents have been implemented in separated files:

- **River** agent (in river.gaml)
    - Aspect:
        - **default**: display with the agent shape
- **db** agent (parent: AgentDB) (in agentCreator.gaml)
  This agent aims at sending request to the Database and initializing section3D agents thanks to the results.
    - Attributes:
        - **SQLITE** (map<string,string>) : all the information to connect to the database
        - **Sqlwl** (string) : the SQL request to get the name of all sections from the database
        - **wlev** (list<float>) : variable that contains the list of H0, received from the DataBase
    - Actions:
        - **load_step** (➔ float) returns the number of steps for which we have data
        - **load_water_levels** (➔ list<float>) : returns the list of water level from database
        - **load_global** : initializes the db agent
        - **init_section3D** : sends a request to get all the points for a section given its name and initialize the section3D agents.
- **section** agent (in section.gaml)
  The section handles everything about hydraulic process in the section, independently to any (3D) representation.
    - Attributes:
        - **section_name** (string) : the name of the section (read form the shapefile)
        - **section_type** (string) : the type of the section (read form the shapefile)
        - **ptsOfSection** (list<point>) : the list of 2D points of the section (note that the top-left point is at {0,0} and the coordinates of every other points are described from this point and in a 2D plan). Note that this list is constant during the simulation. In the model, sections (in our case section3D) are created from a shapefile and their location is a point with absolute coordinates. Note that the top-left point of ptOfSection corresponds to the location of the section. We can thus compute easily the absolute coordinates of all points of ptsOfSection. Note that the coordinate {x,y} of the points of ptsOfSection corresponds respectively to the absolute x and z coordinates. (readers can refer to the code of init_section3D, of agentCreator)

        - **H** (float) : the water level (the height from the sea level)
        - **Q** (float) : the water flow rate
        - **A** (float) : the area of water in a section
        - **R** (float) : the water height from the bottom of the section
        - **s** (float) : the slope of channel bed (this represents the slope from a given section to the previous one)
        - **next_section** (section): the next section (from upstream to downstream)
    - reflex:
        - **testS** (condition: s > 0.0001) : writes a warning message in the console. It represents cases where the steady model is no more adapted to the problem
    - actions:
        - **compute_H** : computes the section agent's H value
        - **compute_A** : computes the section agent's A value
        - **compute_R** : computes the section agent's R value
        - **compute_s** : computes the section agent's s value
    - aspect:
        - **sectio** : display with the agent shape
- **section3D** agent (parent: section) (in section3D.gaml)
  This agent handles everything about the 3D handling of the section agent.
    - attributes:
        - **ptsOfSection3D** (list<point>) : the list of the sections points with absolute and 3D coordinates (constant during the simulation).
        - **WaterptsOfSection3D** (list<list>) : an intermediate variable used to store the 3D points of the water geometry in the section.
        - **river_channel** (geometry) : the geometry of the river channel between the current agent and the next one.
        - **river_water** (list<geometry>) : (only used for display) the 3D shape of the part of the river channel containing water
        - **water_top** (list<geometry>) : (only used for display) the 3D shape of the water surface in the channel.
    - actions:
        - **create_river_channel** : initializes the river_channel from the ptsOfSection3D list. If the section3D is the downstream the river_channel is only the polyline built from the list of points from ptsOfSection3D. Otherwise, it is a polygon built from the list of points from the current agent and the next one (the reverse used in the action is to have the points right order).
        - **updatepoints** : computes the attribute WaterptsOfSection3D. It uses the primitive operator *water_polylines_for* on section shape and a water height to compute the geometry of the water in the section. This geometry is then transformed into a list of list of 3D points. Note that we have a list of list to deal with cases where there is island on the middle of the section.
        - **updategeom** : for the variable WaterptsOfSection3D computed previously of both the current section3D and the one of the next section3D, it computes the geometry of the water_top and water_river.

- aspect:
  - **aspect2D** : display the polyline of ptsOfSection3D
  - **channel** : display the river_channel
  - **default3D** : display the river_channel, river_water and water_top.

- **global** agent (in main.gaml)
  - attributes:
    - **riverFile** (file) : the river shapefile
    - **sectionFile** (file) : the shapefile containing the location of each section
    - **filename** (string) : name of the file in which are stored the results step by step
    - **fileNameVertical** (string) : name of the file where are stored the results step by step, each section on 1 line.
    - **Q0** (float) : the initial and constant water flow
    - **H0** (float) : the water height at the downstream (modified at each step)
    - **end_step** (int) : final step of the simulation
    - **water_levels** (list<float>): the list of all the water height at the downstream
    - **saveAll** (bool) : specify whether we want to store results in files
  - init (cf. Initialization section)
  - reflexes:
    - **computeRiver** (condition: cycle < end_step) : the main reflex of the model:
      - get the H0 value for the current step, modify it if it is too high or low
      - for each section compute the H, A, R, s (from downstream to up stream)
      - update the 3D points for each section3D
      - then update the geometry of each section3D
    - **saveInFile** (condition: saveAll) : save in filename csv file, at each step, the value of the H for each section (from downstream to upstream)
    - **saveResVertical** (condition: saveAll) : save in fileNameVertical csv file, at each step, the H value for each section (from downstream to upstream), one section per line
    - **stop** (condition: cycle>=end_step) : stop the simulation

Initialization

The river agents are created from the associated shapefile. [direct call of the create statement][In init of the global, file: main.gaml]

The section (section3D) agents are created following the two steps:

- First we create the section3D agents from the shapefile, (this provides them their location, that is also the top-left point of their section polyline) [direct call of the create statement][In init of the global, main.gaml file]
- We then get the list of points of the section from the database. [In init of the global, main.gaml file][creation of a DBAgent: agentCreator, call of methods load_step(), load_water_levels() and init_section3D]

Description of the model

The principle of the algorithm is to compute the area of the downstream from the H0 and then compute de water height for each section from downstream to upstream (from steady Q, but taking into account previous H and s).

At each step of the simulation:

- Get the Q0 and H0 value
- For the downstream section,
    - compute A0 (from H0)
    - compute R (as if the shape was a rectangle)
    - compute slope using the formula:

$$n = \frac{1}{Q} * A * R^{2/3} * s^{1/2}$$

So we compute the s with:

$$s = (\frac{n * Q}{A * R^{2/3}})^2$$

- Then for each section (from the downstream to upstream, except downstream), compute the H from the A0 (constant).
    - Compute the H, with:
        - d = the distance between the current section and the following one
        - h = d * next_section.s
        - H = next_section.H + h
    - With the H we can compute A, R, s (as for the downstream section).

Experiment

We provide only one gui experiment: hydro_steadymodelv3.

- parameters:
    - **saveAll** variable (initialize at false)
- outputs:
    - display **morpho**: it displays the section3D agents with the aspect2D aspect and the river agents.
    - display **morpho3D**: it displays the section3D agents with the default3D aspect and the river agents.

Description of the older model: steady model (modelv2)

The same model as the one described above. It seems to contain some bugs, in particular in the management of the sections.

Description of the older model: steady model with A constant (models)

This second model still exists but has not been tested fully.

Hypothesis:

- Steady case (Q is constant)

- The **area A is constant**
- Data hypothesis: sections are ordered

Aim

- Compute the H for each section

Input of the model

- Sections list of points (read from a sqlite database file)
- H0 and Q0 time series

Description of the model

The principle of the algorithm is to compute the area of the downstream from the H0 and then compute de water height for each section from downstream to upstream (with A constant).

At each step of the simulation:

- Get the Q0 and H0 value
- For the downstream, compute the A0 (from H0)
- Then for each section (from the downstream to upstream, except downstream), compute the H from the A0 (constant).

Technical requirements

Models need the SVN in development version of GAMA that should be stabilized soon. In particular, modelv3 uses the new syntax using e.g. list<float>.

Models need a plugin that has been implemented during the Coding Camp. It provides additional operators to the GAML languages with features related to hydro* models:

- water_area_for
    - returns the water area from the section polyline and the water height
- water_level_for
    - returns the water height from the section polyline and the water area
- water_polylines_for
    - returns the water area polygon from the section polyline and the water height. Note that the operator returns a list of list of points, to deal with cases when there is some ground out of the water in the middle of the section (islands for example).

Useful additional tools

To manipulate GIS data:

- OpenJump (http://www.openjump.org/)
- QuantumGIS (http://www.qgis.org/)

To visualize and manipulate SQLite database:

- SQLite Database Browser (http://sqlitebrowser.sourceforge.net/)

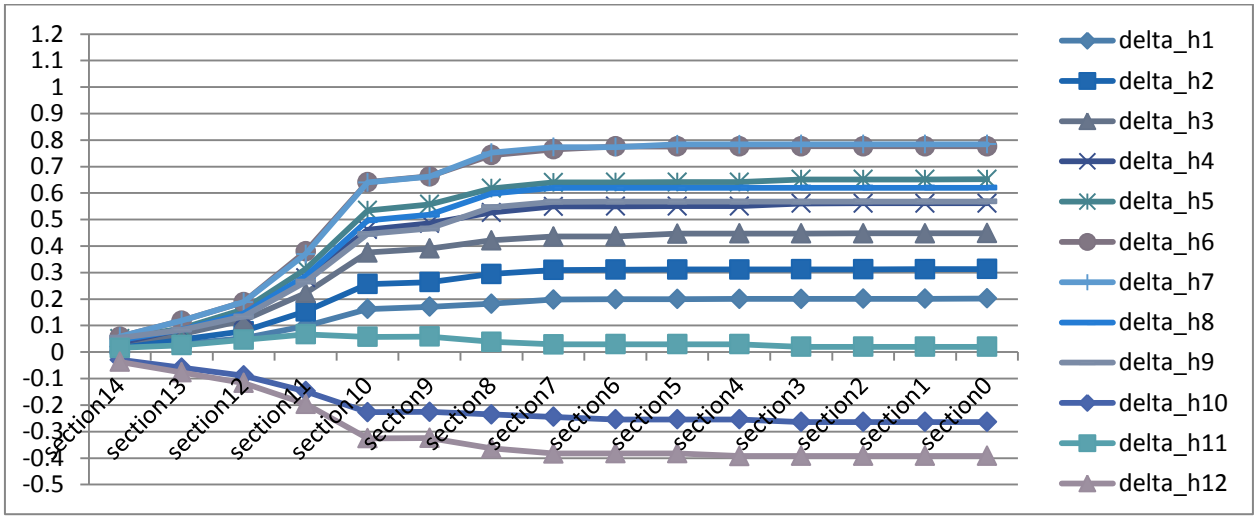
Hydraulic models softwares:

- HEC-RAS (http://www.hec.usace.army.mil/software/hec-ras/)
- ISIS (http://www.halcrow.com/isis/)

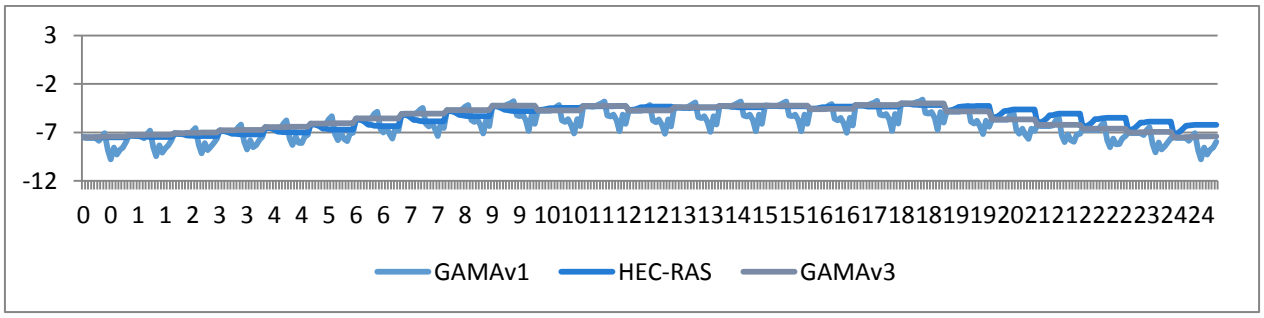
Testing results with modelv2

We have tested the Hydrodynamic model in My Thanh River. There are 10 sections were tested. The results are compared with that the model unsteady on HEC-RAS software (by the U.S. Army Corps of Engineers). The chart in the Figure 1 shows the differences of the water levels of 14 sections between 2 models in 12 hours. We can see that there are not differences in 3 firsts sections and at 1h and 11h, but there are not fixed with the others cases. That cause by we compared the models with the unsteady model of HEC-RAS. We will continue develop the unsteady model in GAMA to fix that.

We modelv2 we get following results:



Results comparison of HEC and GAMAv3 (cf. HydroResult_modelv3.xlsx file)



Conclusion

The modelv3 is strangely very stable along the river. Additional works are needed to compare modelv3 and the HEC-RAs results. The implemented model does not seem to reproduce the behavior we can observe on HEC. In particular depending on the simulation step, the water height can either increase or decrease along the river, which cannot be observed on our model.

In addition, the various data that are compared should be checked (from GAMA we get the H value, we should be sure that we also get this value from HEC-RAS model). Moreover, even the H0 does not seem to be the same on GAMA model and HEC-RAS models.