WASA

Tests part

Here we will describe all the tests we thought interesting to do. There are 16 cases in total. Let's list which variables of the folder ***do.dot*** we will modify. There are 12 variables which are :

- doreservoir *(line 14)*
- doacudes *(line 15)*
- dotrans *(line 20)*
- dohour *(line 21)*
- dointc *(line 26)*
- dt *(line 30)*
- dosediment *(line 31)*
- river routing *(line 34)*
- optional: load state of storages from files (if present) at start *(line 36)*
- optional: save state of storages to files after simulation period *(line 37)*
- dosnow *(line 38)*
- doirrigation *(line 39)*

The 17 cases are described in the following list and table with the name of the test folder.

- *TestInput1* : the basic case
- *TestInput2* : the basic case but with hourly version instead of da
- *TestInput3* : reservoir are taken into account
- *TestInput4* : reservoir and hourly version
- *TestInput5* : reservoir without the doacudes mode with daily data
- *TestInput6* : reservoir without the doacudes mode with hourly data
- *TestInput7* : transpiration is taken into account
- *TestInput8* : reservoir + modified bucket
- *TestInput9* : sediment and reservoir are taken into account
- *TestInput10* : sediment and reservoir are taken into account with an hourly mode
- *TestInput11* : sediment and reservoir are taken into account with Muskingum & ss transport (river routing)
- *TestInput12* : sediment and reservoir are taken into account with Muskingum & bedload modelling (river routing)
- *TestInput13* : sediment and reservoir are taken into account with save state and load state options
- *TestInput14* : sediment and reservoir are taken into account without save state and load state options
- *TestInput15* : snow, reservoir, sediment, irrigation and transpiration are taken into account
- *TestInput16* : irrigation, sediment and reservoir are taken into account

It is important to specify what is the « basic case ». Let's show the ***do.dat*** of this case. The line in bold are the variables which will change during our test. The others will stay the same and keep the value they have in the ***do.dat*** of the basic case as below.

(1) Parameter specification for the WASA Model (SESAM-Project), generated with R-Package lumpR function db_wasa_input() version 3.0.48
(2) .\Input\
(3) .\Output\
(4) 2000 //tstart (start year of simulation)
(5) 2000 //tstop (end year of simulation)
(6) 1 //mstart (start month of simulation [optional: <space>start_day])
(7) 12 //mstop (end month of simulation [optional: <space>end_day])
(8) 2 //no. of sub-basins
(9) 12 //no. of combinations of sub-basins, landscape units, terrain components (TC-instances)
(10) 2 //total no. of landscape units in study area
(11) 6 //total no. of terrain components (types) in study area
(12) 2 //total no. of soil components in study area
(13) 2 //total no. of vegetation units in study area
(14) .f. //doreservoir: do reservoir calculations
(15) .t. //doacudes: include calculations for small reservoirs
(16) .t. //dolattc: do latflow between TCs
(17) .f. //doalllattc: route latflow compeletely to next downslope TC
(18) .t. //dolatsc: do latflow within TCs (surface runoff)
(19) .t. //dolatscsub: do latflow within TCs (subsurface runoff)
(20) .f. //dotrans: do water transpositions betwen sub-basins
(21) .f. //dohour: do hourly version
(22) 0 //legacy: scenario: choose scenario (0:less rain (ECHAM), 1:no trend, 2:more rain (Hadley))
(23) 0 //legacy: krig: type of precipitation interpolation (0:OK, 1:EDK, 2:EDKxyz, 3:csimabsed3, 4:csimreled3, 5:csimreled1, 6:csimabsed1, 7:statdata, 8:statdatacon, 9:gerstdatacon, 10:gerstdata, 11:ok_mean1cell)
(24) 15.0 //kfkorr: hydraulic conductivity factor (for daily model version) (kfkorr)
(25) 0.30 //intcf: interception capacity per unit LAI [mm]
(26) 0 //dointc: type of interception routine (0:simple bucket, 1:modified bucket)
(27) .t. //doscale: do scaling due to rainfall interpolation ?
(28) .f. //domuncell: for muni/ezg-nocell-version, use rainfall input derived from cells ? (change kf_calib.dat !)
(29) 1. //sensfactor: factor for sensitivity studies
(30) 24 //dt: time step in [hours]
(31) .f. //dosediment
(32) 3 //No. of soil grain size classes
(33) 1 // type of sediment transport model at the hillslope
(34) 1 //type of water / sediment model in the river: (1) old routing, (2) Muskingum & ss transport, (3) Muskingum & bedload modelling
(35) 1 //type of sediment model in the reservoir: choose sediment transport equation (1:Wu et al., 2000; 2:Ashida and Michiue, 1973; 3: Yang, 1973 and 1984; 4: Ackers and White, 1973)
(36) .t..f. //OPTIONAL: load state of storages from files (if present) at start; optional second flag: allows the model to append to existing output files, default is .f.
(37) .t. .f. //OPTIONAL: save state of storages to files after simulation period; optional second flag: determines if the model states are saved (and overwritten) at the end of each simulation year, default is .t.
(38) .f. // dosnow
(39) .f. //doirrigation

1. First set

Folder → / ↓ Options	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
reservoir	.f.	.f.	**.t.**	**.t.**	**.t.**	**.t.**	.f.	**.t.**	**.t.**	.t.	**.t.**	.t.	**.t.**	**.t.**	**.t.**	**.t.**
doacudes	.t.	.t.	.t.	.t.	**.f.**	**.f.**	.t.	.t.	.t.	.t.	.t.	.t.	.t.	.t.	.t.	.t.
dotrans	.f.	.f.	.f.	.f.	.f.	.f.	**.t.**	.f.	.f.	.f.	.f.	.f.	.f.	.f.	**.t.**	.f.
dohour	.f.	**.t.**	.f.	**.t.**	.f.	**.t.**	.f.	.f.	.f.	.t.	.f.	.f.	.f.	.f.	.f.	.f.
dt	24	**1**	24	**1**	24	**1**	24	24	24	1	24	24	24	24	24	24
dointc	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0
sediment	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	**.t.**	.t.	**.t.**	.t.	**.t.**	**.t.**	**.t.**	**.t.**
river routing	1	1	1	1	1	1	1	1	1	1	**2**	3	1	1	1	1
option : load state	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	**.t.**	**.f.**	.t..f.	.t..f.
option : save state	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	**.t.**	**.f.**	.t..f.	.t..f.
snow	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	**.t.**	.f.
irrigation	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	**.t.**	**.t.**

Table 1: *Description of the 16 cases tested. Changes from the basic case are emphasized by the blue cells.*

Note : with the hourly mode some out files still give the daily data. Indeed, all the daily file do. We can also notice that some .out files do not depend on the time but also all the .stat and .stat_start.

For the cases TestInput10 and TestInput12 almost all the files are empty. This is why there are in light blue in the table.

Analysis : Regarding sediments.

- ➡ with sediment on new outfiles: « river_sediment_total.out », « susp_sediment_storage.stat », « susp_sediment_storage.stat_start », « daily_sediment_production.out » → TestInuput9
- ➡ file « sediment_production.out » is present in TestInput10 out files but not in TestInput9
- ➡ in TestInput11 we get out files that we did not have for the two previous cases which are « sediment_storage.stat » and « sediment_storage.stat_start » and we don't have the file « sediment_production.out » that we had in TestInput10
- ➡ **why ?**
- ➡ maybe the sediment_storage files appear because we change the river routing ?

Out files which have hour dependency :

- gw_recharge.out
- gw_discharge.out
- gw_loss.out
- qhorton.out
- potetranspiration.out
- water_subbassin.out
- total_overlandflow.out
- subsurface_runoff.out
- actetranspiration.out

Analysis : Regarding the file « lake_storage.out »

➡ does not exist for TestInput5 and TestInput6

➡ all the values of the file are empty for TestInput9, TestInput11, TestInput13, TestInput14 and TestInput16

➡ the initial condition file with the same name has values different from 0 (folder init_conds)

➡ **why ?**

Analysis : Regarding the file « gw_loss.out »

➡ all the values of the file are empty for each case

➡ exception : the file is empty for TestInput12

➡ **why ?**

Physical analysis : Regarding the hourly data for the case TestInput2

➡ The file « potetranspiration.out » presents the maximum of evapotranspiration possible by hour during each day and « daily_potetranspiration.out » presents the maximum of evapotranspiration possible by day. We can notice that the sum of the hourly data for one day in « potetranspiration.out » is different (but close) from the corresponding daily value on « daily_potetranspiration.out ». Exemple :

- day 1, sub-basin 15 : $E_{sum} = 2{,}448\ mm/d$ and $E_{daily} = 2{,}453\ mm/d$
- day 2, sub-basin 15 : $E_{sum} = 1{,}536\ mm/d$ and $E_{daily} = 1{,}527\ mm/d$

➡ **what is the physical explanation ?**

➡ For the files « actetranspiration.out » and « daily_astetranspiration.out » we get the same value (between the sum of the hourly values and the corresponding daily value) which is consistent.

➡ Comparing « actetranspiration.out » and « potetranspiration.out » we can see that some actual values are higher than the potential ones. Exemple :

- day 1, hour 10, sub-basin 15 : $E_{pot} = 0{,}102\ mm$ and $E_{act} = 0{,}126\ mm$

➡ **what is the explanation ?**

Physical analysis : Regarding reservoir storage in TestInput3 (probably in the other cases too)

➡ In the file « reservoir_storage.stat_start » we can see that we have negative volumes.

➡ **what is the explanation for that ?**

Analysis : Regarding the files « soil_moisture.stat » and « soil_moisture.stat_start »

➡ We can note that for some lines on the files there is a difference between the data for the water content (wc). Exemple :

- In TestInput4, line 4 : $wc_{stat} = 751{,}95\ mm$ and $wc_{stat\_start} = 751{,}87\ mm$

➡ **is there an explanation to that ?**

2. Second set

Thanks to all these analysis we did after the first set of cases, we do some changes on the code to solve the issue we encountered.
First, the most important thing is not to get empty output files for the cases 10 and 12. The presence of empty files was due to a bug for case 10 and missing files for case 12. We also need a new executable file. We call this new one ***wasaP.exe*** and so we call the initial one ***wasaI.exe***. These names are just here to identify the executable file. It is important to notice that considering the command wrote on the ***Run_WASA.bat*** file, we need to remove the letter I or P of the executable we want before running the code.

Now, the new task is, for each case :

- in the ***do.dat*** file, putting true for the option regarding the load state (ligne 36)
- running the code with ***wasaP.exe***
- copying the ****.stat_start*** files get in the Output folder to the Input folder and renaming them to have to extension ****.stat***
- running again the code with ***wasaT.exe***

The executable ***wasaT.exe*** is the result of changes in the code to take the ****.stat*** files from the Input folder.

The new cases are reported on Table 2. We can see in green the changes on the ***do.dat*** file from the previous set. They are just on the line of the load state option as we explained before.

Folder → / **↓ Options**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
reservoir	.f.	.f.	**.t.**	**.t.**	**.t.**	**.t.**	.f.	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**
doacudes	.t.	.t.	.t.	.t.	**.f.**	**.f.**	.t.	.t.	.t.	.t.	.t.	.t.	.t.	.t.	.t.	.t.
dotrans	.f.	.f.	.f.	.f.	.f.	.f.	**.t.**	.f.	.f.	.f.	.f.	.f.	.f.	.f.	**.t.**	.f.
dohour	.f.	**.t.**	.f.	**.t.**	.f.	**.t.**	.f.	.f.	.f.	**.t.**	.f.	.f.	.f.	.f.	.f.	.f.
dt	24	**1**	24	**1**	24	**1**	24	24	24	**1**	24	24	24	24	24	24
dointc	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0
sediment	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**
river routing	1	1	1	1	1	1	1	1	1	1	**2**	**3**	1	1	1	1
option : load state	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	**.t.**	.t..f.	.t..f.
option : save state	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	.t..f.	**.t.**	**.f.**	.t..f.	.t..f.
snow	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	**.t.**	.f.
irrigation	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	.f.	**.t.**	**.t.**

Table 2: *Description of the new 16 cases tested. Changes from the basic case are emphasized by the blue cells. Changes from the previous set of cases are in green.*

For the cases we get different ***.stat_start*** files. Indeed, the files we get are listed in Table 3.

Folder → / ↓ Files	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
gw_storage	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
intercept_storage	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
interflow_storage	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
reservoir_storage			✘	✘	✘	✘		✘	✘	✘	✘	✘	✘	✘		✘
river_storage	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
soil_moisture	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
lake_storage	✘	✘	✘	✘			✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
susp_sediment_storage									✘	✘	✘	✘	✘	✘		✘
sediment_storage											✘	✘				

Table 3: *Description of the *.stat files present in the Input folder (after the copy-paste and the name modification) for the 16 cases tested.*

Analysis : Regarding table 3

- ➡ The file « reservoir_storage.stat » is not present for the cases 1, 2, 7 and 15. It seems consistant not to have this file for the cases 1, 2 and 7 because the reservoir option in off.
  - ► **why don't we have this file for the case 15 while we have reservoir on? Is it because of the snow?**
- ➡ The file « lake_storage.stat » is not present for cases 5 and 6. This is probably due to have putting the option doacudes false.
- ➡ The file « susp_sediment_storage.stat » appears when we put the sediment option to on. There is still something strange with the case 15 because we don't have this file.
  - ► **case 15 does not work, see below**
- ➡ The file « sediment_storage.stat » is present just for the cases 11 and 12. Probably due to the changes on the river routing.

Tests regarding case 15 :

We notice that if we delete all the files in the Ouput folder of the case 15, and then we run the code, just a file named « parameter » appears in the folder. So this case does not work. This is why we decide to do some tests that we can see on table 4.

Folder → **↓ Options**	**15**	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**
reservoir	**t**	**t**	**t**	**t**	f	**t**	**t**	**t**	**t**	**t**	**t**	**t**
dotrans	**t**	f	**t**	**t**	**t**	**t**	**t**	f	f	**t**	**t**	f
sediment	**t**	**t**	**t**	**t**	**t**	f	**t**	**t**	**t**	f	f	**t**
snow	**t**	**t**	f	**t**	**t**	**t**	t	**t**	f	f	**t**	f
irrigation	**t**	**t**	**t**	f	**t**	**t**	t	f	**t**	**t**	f	f
Results ?	❌	❌	❌	❌	❌	❌	❌	❌	❌	❌	❌	✅

Table 4: *Description of the tests made to understand what are the issues in the case 15.*

Documentation and research on Unit Testing

The aim of this section is to learn more about Unit Testing (UT) and to see what is the best way for us to apply this method to our study.

1. Focus on UT

The unit can be what we want but this is better when it is small. Indeed, it gives us a better overview of how the code is performing. Also, with a small unit the test will occur faster than with a big unit. [1]
Different techniques exist such as [2]:

- Black Box Testing : user interface, input and output are tested.
- White Box Testing : each function behavior is tested. (usual one [3])
- Gray Bow Testing : « used to execute tests, risks and assessment methods. »

Usually, the unit testing is performed by the software developers themselves. In other cases, this is done by an other software tester which has to have the code and know the architecture. [3] It has to be written in the same language as the tested software. Some tips from [3] (copied and pasted) :

- Create a proper unit test plan. [If not documented, at least in your head.]
- Find a test automation tool / framework for your language.
- Create test cases focusing on areas that impact the behavior of the system the most.
- Isolate the development environment from the test environment.
- Use test data that is close to that of production.
- Write test cases that are independent of each other. For example, if a class depends on a database, do not write a case that interacts with the database to test the class. Instead, create an abstract interface around that database connection and implement that interface with a mock object.
- Make sure you are using a version control system to keep track of your test scripts.
- In addition to writing cases to verify the behavior, write cases to ensure the performance of the code.
- Perform unit tests continuously and frequently.

2. Continuous integration

By using continuous integration [4] :

- we always know the latest stable version of our software
- we have instant feedback if a developer‘s work in progress breaks the stable version
- we can automatically test different setups

⇒ unit tests are executed to determine the success of a build and are automated

3. Tools

Jenkins seemed to be a good solution but Java is necessary to use it. We want something easy to use by everyone so this is not a good option because this is a really specific software (certain space on the laptop, Java and downloading Jenkins is needed).

Other options [5] :

- CTest in CMake permit to check output and return value of external processes
- cygwin or msys on Windows : are softwares to use Unix application in Windows => test possible just for C projects
- pexpect from Python : can be used for automates software testing. Should work on any platform supporting the standard pty module. => seems less restrictive option but check if it is possible to do something with comparison between out files

References :

[1] https://smartbear.com/learn/automated-testing/what-is-unit-testing/

[2] https://www.tutorialspoint.com/software_testing_dictionary/unit_testing.htm

[3] https://softwaretestingfundamentals.com/unit-testing/

[4] https://www.informatik.hu-berlin.de/de/forschung/gebiete/wbi/teaching/archive/ws_1819/sp_brettspiel/07_unittests_ci.pdf

[5] https://softwareengineering.stackexchange.com/questions/202643/cppunit-for-unit-testing-executable-files