# Dev Doc – Development Advices

MATLAB HEC-RAS Interface Documentation, IMTLD

**Last update: 9/7/2020 1:01:00 PM – version BETA 1_0_1**

This paper aims to give you all information and avenues already explored, which should help you editing the script. To understand well this paper, I recommend reading the user guide before.

There is not always logical link between one paragraph and another, because this paper is a listing of ideas.

To get information about a specific function or object, write help <function or object> in the MATLAB command window.

## Table of contents

# Notes

## Misuse of language

Strings and character vectors are not the same object in MATLAB, but character vectors are often called strings in this documentation.

## Way to code

Variable names are clear but long, thanks to autocomplete functionality of VS Code.

If you must edit this code, choose an editor which can perform autocompletion.

## Known limitations of the interface

HEC-RAS allows that different boundary conditions of the same name exists on a XS. This is NOT supported by the MATLAB HEC-RAS interface, and will not throw an error.

**Note:** multiple gates on a XS are supported because they have a unique gate number, e.g. Gate #1 and Gate #2

If you want to bypass this limitation, you will have to find a way to detect a difference between these two conditions. A priori, there is no way to distinguish them, except a comparison of the data they contain. Which is not reliable, because these data can be modified by MATLAB or by HEC-RAS depending on your script.

## Code internal documentation

There is documentation in every function of the code. Including pseudocode information if necessary.

To quickly access this documentation, write the function or object name in the MATLAB command window, and then press F1.

# Debug methodology

This methodology is used to make the interface working together with HEC-RAS and understand how to solve encountered issues.

## First part: the tools

You will run a lot of times your HEC-RAS project, so configure it to keep the computation as much quick as possible. A project as fast as shown in the video tutorial 1 – Simple automation example is a good basis.

To know which values MATLAB must edit in Ras, a code comparator, which support all text file extension. A nice one is the compare tool in MATLAB > HOME menu > COMPARE button. The VS Code (free software) compare tool works well too, but it is less handy.

## Second part: detect necessary edits to do in Ras files

### First step: capture Ras files states at different moments

To do the three practical works explained below, this is safer to make a HEC-RAS reference project, and copy it in three different folders, one for each practical work.

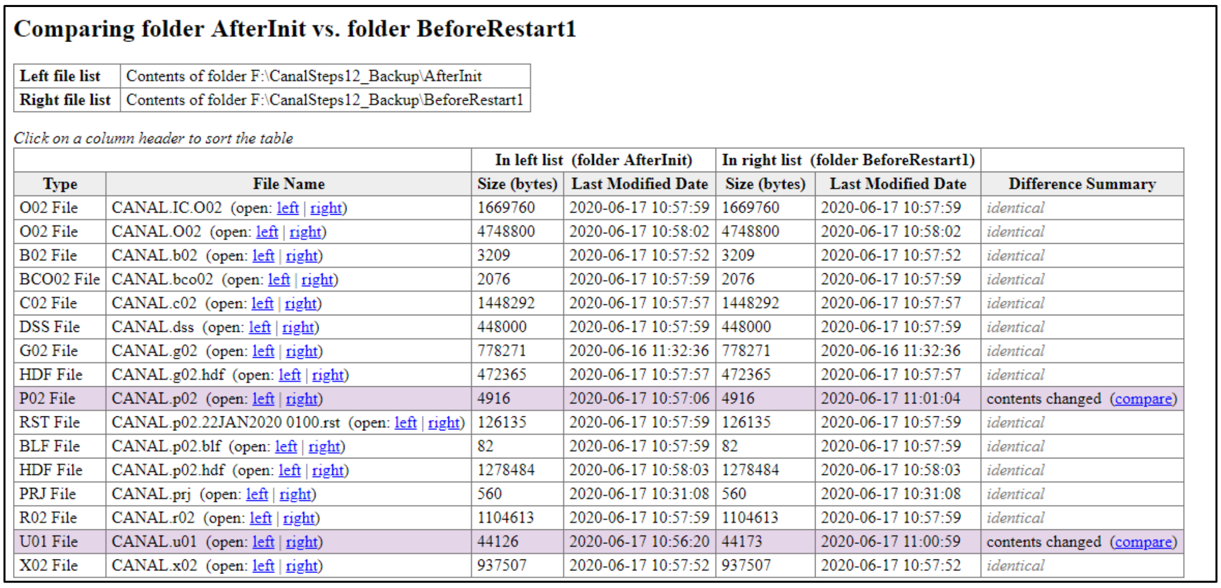
a. In settings.ini > debug=1; run MATLAB HEC-RAS interface in MATLAB command window. Always backup when prompted. There is an advanced backup system which avoid overwriting of backups, between different time steps.

b. Open HEC-RAS and run the same simulation, but pause HEC-RAS at each time step, exactly like what MATLAB has done during [a.]. Watch the dev tutorial 1 – How to manually pause and restart a Ras simulation to learn how to do so.

c. Finally, run the same simulation in HEC-RAS (like in [b.]) but straight, without pause. This methodology is the one you follow, when you run a usual simulation in HEC-RAS, without any automation.

### Second step: comparison of results

After that, compare every file which belong to the same step of the Ras simulation. Also compare every file which belong to two successive steps of the Ras simulation. You should obtain results similar with results presented in the screen capture below.



**Comparing folder AfterInit vs. folder BeforeRestart1**

| | |
|---|---|
| **Left file list** | Contents of folder F:\CanalSteps12_Backup\AfterInit |
| **Right file list** | Contents of folder F:\CanalSteps12_Backup\BeforeRestart1 |

*Click on a column header to sort the table*

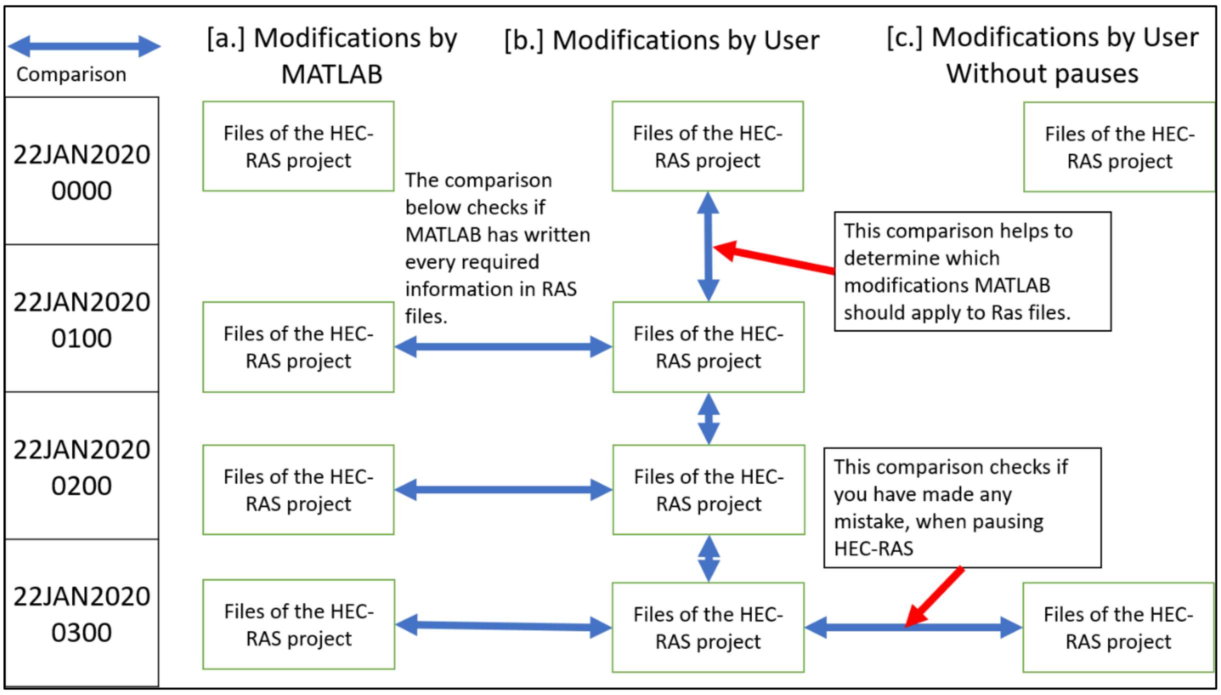
| | | **In left list (folder AfterInit)** | | **In right list (folder BeforeRestart1)** | | |
|---|---|---|---|---|---|---|
| **Type** | **File Name** | **Size (bytes)** | **Last Modified Date** | **Size (bytes)** | **Last Modified Date** | **Difference Summary** |
| O02 File | CANAL.IC.O02 (open: left \| right) | 1669760 | 2020-06-17 10:57:59 | 1669760 | 2020-06-17 10:57:59 | *identical* |
| O02 File | CANAL.O02 (open: left \| right) | 4748800 | 2020-06-17 10:58:02 | 4748800 | 2020-06-17 10:58:02 | *identical* |
| B02 File | CANAL.b02 (open: left \| right) | 3209 | 2020-06-17 10:57:52 | 3209 | 2020-06-17 10:57:52 | *identical* |
| BCO02 File | CANAL.bco02 (open: left \| right) | 2076 | 2020-06-17 10:57:59 | 2076 | 2020-06-17 10:57:59 | *identical* |
| C02 File | CANAL.c02 (open: left \| right) | 1448292 | 2020-06-17 10:57:57 | 1448292 | 2020-06-17 10:57:57 | *identical* |
| DSS File | CANAL.dss (open: left \| right) | 448000 | 2020-06-17 10:57:59 | 448000 | 2020-06-17 10:57:59 | *identical* |
| G02 File | CANAL.g02 (open: left \| right) | 778271 | 2020-06-16 11:32:36 | 778271 | 2020-06-16 11:32:36 | *identical* |
| HDF File | CANAL.g02.hdf (open: left \| right) | 472365 | 2020-06-17 10:57:57 | 472365 | 2020-06-17 10:57:57 | *identical* |
| P02 File | CANAL.p02 (open: left \| right) | 4916 | 2020-06-17 10:57:06 | 4916 | 2020-06-17 11:01:04 | contents changed (compare) |
| RST File | CANAL.p02.22JAN2020 0100.rst (open: left \| right) | 126135 | 2020-06-17 10:57:59 | 126135 | 2020-06-17 10:57:59 | *identical* |
| BLF File | CANAL.p02.blf (open: left \| right) | 82 | 2020-06-17 10:57:59 | 82 | 2020-06-17 10:57:59 | *identical* |
| HDF File | CANAL.p02.hdf (open: left \| right) | 1278484 | 2020-06-17 10:58:03 | 1278484 | 2020-06-17 10:58:03 | *identical* |
| PRJ File | CANAL.prj (open: left \| right) | 560 | 2020-06-17 10:31:08 | 560 | 2020-06-17 10:31:08 | *identical* |
| R02 File | CANAL.r02 (open: left \| right) | 1104613 | 2020-06-17 10:57:59 | 1104613 | 2020-06-17 10:57:59 | *identical* |
| U01 File | CANAL.u01 (open: left \| right) | 44126 | 2020-06-17 10:56:20 | 44173 | 2020-06-17 11:00:59 | contents changed (compare) |
| X02 File | CANAL.x02 (open: left \| right) | 937507 | 2020-06-17 10:57:52 | 937507 | 2020-06-17 10:57:52 | *identical* |

*Figure 1 - Comparison after initialization (first step) and before first restart (2d step)*

In the picture above, we capture the changes that were made to files by the user between two simulation steps.

## What should you compare?

The plan below synthetizes the methodology. According to the information gathered during First step: capture Ras files states at different moments.



Comparison
[a.] Modifications by MATLAB
[b.] Modifications by User
[c.] Modifications by User Without pauses
22JAN2020 0000
22JAN2020 0100
22JAN2020 0200
22JAN2020 0300
Files of the HEC-RAS project
The comparison below checks if MATLAB has written every required information in RAS files.
This comparison helps to determine which modifications MATLAB should apply to Ras files.
This comparison checks if you have made any mistake, when pausing HEC-RAS

*Figure 2 - Plan of comparisons, which are useful for debug*

## Third part: coding in MATLAB edits to do to Ras files

### First step: code editing

You can now edit the code to make the interface able to automate HEC-RAS. To do so, try to use functions already written, which are in the folder containing the code of the interface, e.g. BETA 1_0_1 MATLAB HEC-RAS interface > MATLAB Code.

It is recommended to create a CUSTOM version if you edit any function in the MATLAB code, to be aware that this version of the interface is not the one you downloaded from our website. See the Create your own custom version paragraph of this paper.

Here are some advices of which part of the scripts you should edit:

- Master.m function, the part of code, which is inside the for loop, especially code lines around the calls to update_param_first function
- Functions update_param_first and param_delete are the only one used to write in HEC-RAS file the modifications related to pause and restart.

### Second step: check results

Check modifications you performed on the code. To do so, go back to Second part: detect necessary edits to do in Ras files and check thanks to comparisons that your scripts behave in the good way.

## Coding methodology

### Add new features

First, you should read the Dependencies between parameters paragraph to understand how you can edit the MATLAB HEC-RAS interface and which part of the script you should edit. Do not forget to create a CUSTOM version, cf. Create your own custom version.

Then, take time to understand and read documentation of functions and class. Note that they are sorted in a directory tree. The goal of this tree is to help you finding quickly which functions you need to call, edit and so on. To save time, try to use as much as possible the functions which are already in the interface. And do not hesitate to contact us if you have any suggestion. We thank you in advance for your help.

Finally, to help you coding, the next paragraphs present methodologies and advices for concrete cases, when a script edit is required.

### Create your own custom version

There are three types of "official" versions of the interface:

- ALPHA (not available for download, this type of versions was used during the beginning of the design of the interface).
- BETA (unstable)
- RELEASE (stable)

You can create a new version type very easily, your folder version name must simply follow this pattern, to make the launcher script able to recognize it as an installed version:

*Let X an integer between 0 and 9, let Y an integer between 0 and 999*

<the name of your type of version> Y MATLAB HECRAS interface

<the name of your type of version> X_X_X MATLAB HECRAS interface

By default, launcher launch the latest version, whatever its type, as long as it is a string of characters (A-Za-z). So, if you want to launch your custom version by default, choose a version number higher than the last "official" version.

**Note:** X_X_X is computed as the XXX number, e.g. 1_0_1 is computed as 101, so 102 is a higher version number.

**Note 2:** MATLAB 2020 does not accept folder names which contains '.', so a folder named 1.0.1 is forbidden, despite it is handier.

## Make an upgrade of MATLAB or HEC-RAS supported version

The script is designed to work with MATLAB 2020 and HEC-RAS 5.0.7. It is likely to not work with later versions of theses software.

To make this script, I used documents which were written for previous versions of MATLAB and HEC-RAS. So, I can give you some advices.

For a later version of HEC-RAS, the main idea is to use the

Debug method in a different context.

First, we will make a simulation manually, with pause and restart of HEC-RAS and we will backup the files of HEC-RAS before and after each time step.

The table below synthetize comparisons of HEC-RAS files to do. To choose which modifications you should do to the code.

Manual Ras simulation is with pauses and restart.

| What | This | Compared with this |
|---|---|---|
| **Manual Ras simulation** | Latest Ras version supported by the interface | New Ras version |
| **Automated Ras simulation** | Latest interface version, Latest Ras version supported by the interface | Latest interface version, new Ras version<br><br>**Note:** errors should occur, write them down for later use |
| **Manual VS. Automated Ras simulation** | Manual simulation in the new Ras version | New edited MATLAB script, new Ras version |

Yet, if MATLAB language has been modified, you should use the internal MATLAB documentation to make this script compatible with the new MATLAB version. I didn't encounter any problem when I designed this script, except that the dlmread function was not recommended, and was used in the article Controlling HEC-RAS using MATLAB, cf. Reference sources paper.

## Function dependencies

There is a lot of dependencies between functions of the script. That is why, editing a low-level function, i.e. which is called by a lot of other functions, can causes a lot of issues which can be hard to fix.

### Analysis methodology to obtain a dependencies graph for BETA 1_0_1 version

The MATLAB internal documentation page Analyze Project Dependencies explains well how to make a dependencies graph.

I created a MATLAB project which had as parent folder, the folder which contains the launcher script. The GRAPHML file, from which an extract is presented in the screenshot below, is available in the resources folder.

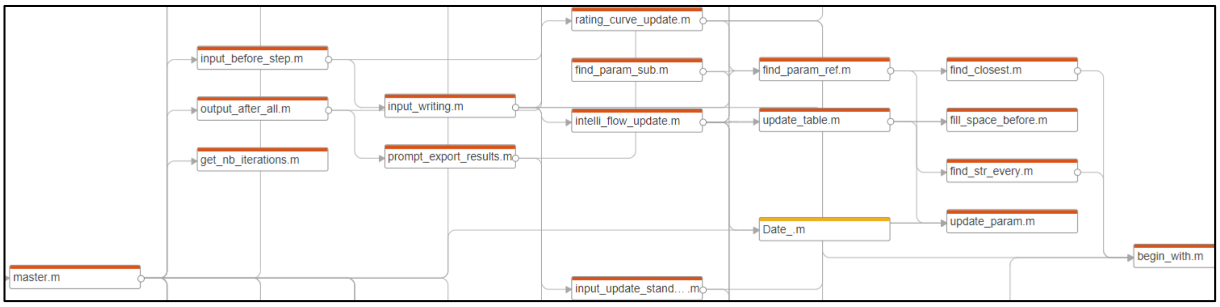


*Figure 3 - Sample of the Dependencies graph of version BETA 1_0_1*

In the sample above, we can see the numerous functions called to understand the input_list returned by the input_scripts.

## How to improve detection of errors occurring HEC-RAS side

Quotation of the Troubleshooting paper:

*If HEC-RAS encounters an error during simulation launch, it will not always throw an error to MATLAB. In the case it does not throw an error, there is different possible behaviors:*

*[…]*

*The HEC-RAS Controller object can stay in an infinite loop. Often, HEC-RAS has written an error message in the TXT file "[…] compute msgs", but in the case of a full automation. MATLAB will not be aware of that error and will endlessly wait for HEC-RAS.*

It is hard to detect an error in the compute msgs text file because the word error is not written, there is only status messages explaining what is missing. But a priori, there is no recognizable word which can be understood by MATLAB.

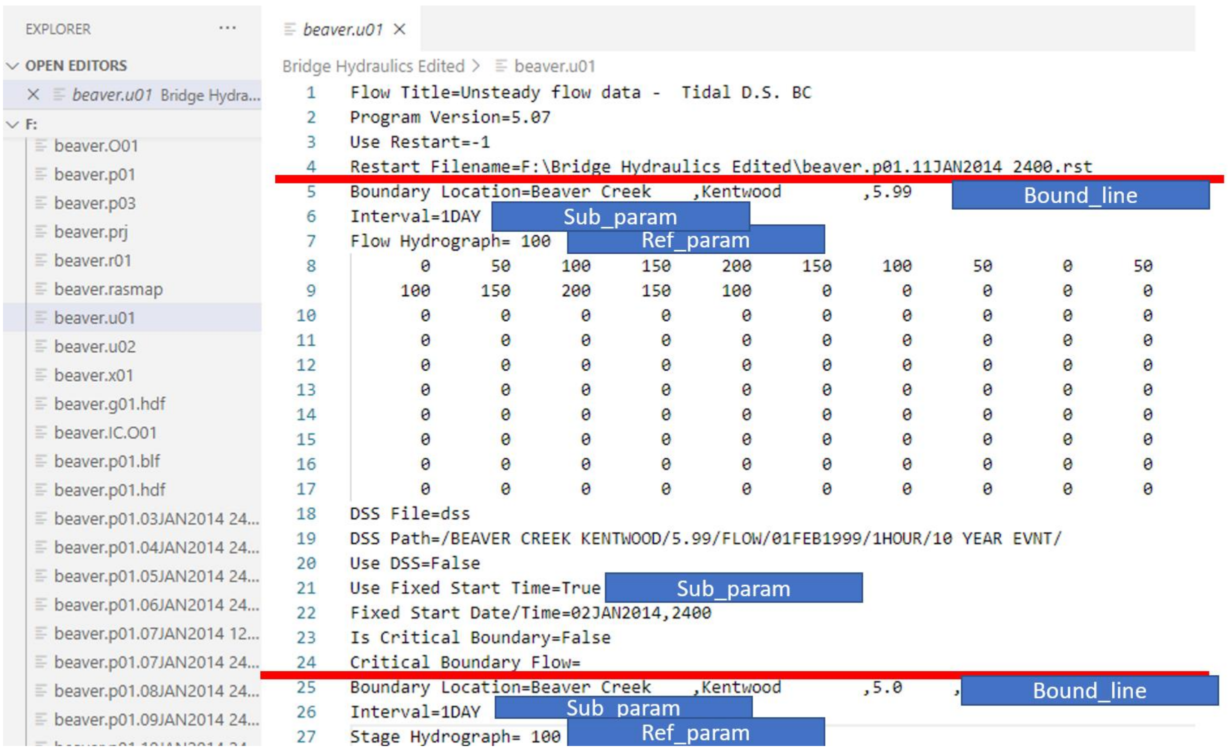
The error detection integrated in the interface is based on a number of lines measurement. Because if the simulation is successful, the compute msgs text file will have a number of line higher than 100, and if an error occurs at the beginning, the file length will be lower than 50.

But if an error occurs after some simulation steps, and the HEC-RAS Controller stays in an infinite lop, it will not be detected by MATLAB, and MATLAB will wait endlessly for HEC-RAS.

There is perhaps a solution, if you measure the average simulation time of a time step and throw an error to MATLAB if a step takes too much time. But it can work only for a specific project.

## Dependencies between parameters

HEC-RAS allows that different boundary conditions exists on a XS. If they have different names, this is supported by the MATLAB HEC-RAS interface. But they are not saved in the same order as they are displayed in Ras GUI. The easiest way to understand that is to look at Ras unsteady flow file structure :


EXPLORER
OPEN EDITORS
beaver.u01 Bridge Hydra...
F:
beaver.O01
beaver.p01
beaver.p03
beaver.prj
beaver.r01
beaver.rasmap
beaver.u01
beaver.u02
beaver.x01
beaver.g01.hdf
beaver.IC.O01
beaver.p01.blf
beaver.p01.hdf
beaver.p01.03JAN2014 24...
beaver.p01.04JAN2014 24...
beaver.p01.05JAN2014 24...
beaver.p01.06JAN2014 24...
beaver.p01.07JAN2014 12...
beaver.p01.07JAN2014 24...
beaver.p01.08JAN2014 24...
beaver.p01.09JAN2014 24...
beaver.u01
Bridge Hydraulics Edited > beaver.u01
Bound_line
Sub_param
Ref_param
Sub_param
Bound_line
Sub_param
Ref_param

```
Flow Title=Unsteady flow data -  Tidal D.S. BC
Program Version=5.07
Use Restart=-1
Restart Filename=F:\Bridge Hydraulics Edited\beaver.p01.11JAN2014 2400.rst
Boundary Location=Beaver Creek     ,Kentwood        ,5.99
Interval=1DAY
Flow Hydrograph= 100
       0       50      100      150      200      150      100       50        0       50
     100      150      200      150      100        0        0        0        0        0
       0        0        0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0        0        0
DSS File=dss
DSS Path=/BEAVER CREEK KENTWOOD/5.99/FLOW/01FEB1999/1HOUR/10 YEAR EVNT/
Use DSS=False
Use Fixed Start Time=True
Fixed Start Date/Time=02JAN2014,2400
Is Critical Boundary=False
Critical Boundary Flow=
Boundary Location=Beaver Creek     ,Kentwood        ,5.0        ,
Interval=1DAY
Stage Hydrograph= 100
```

*Figure 4 - Edited screenshot of an extract of a unsteady flow Ras file*

**Definitions**

- A ref_param is, according to the known limitation of the interface, defined as a parameter which is unique for a given cross section.

- A sub_param is defined as a parameter which is unique for a given ref_param.

In the example above, if Stage Hydrograph and Flow Hydrograph were two boundary conditions applied on the 5.99 XS, it will be two identical params named Interval. But, as far as we know the ref_param which is linked to each of these parameters, we can easily find them and edit them without any risk of confusion.

# Description of noteworthy MATLAB objects

## Script launcher

If you are using the interface in a normal way, the launcher is called before each launch of an interface version.

**Cleanup**

Delete global variables, close files.

Remove from MATLAB search path, previously launched interface versions.

**Warning:** it restores MATLAB search path to factory settings, make sure it does not cause any trouble for you. Cancel any addpath call made before the call to launcher script

**Init**

Prompt user which version to launch, call launching functions depending on the user answer.

**Launching functions**

If it is an ALPHA version, launcher change the current MATLAB folder and copy every user edited file into the ALPHA version code folder.

Else (RELEASE, BETA or CUSTOM version), launcher add to path the version code folder and every subfolder.

Then launcher call the master function of the given version.

## Folder Installation

Contains a set of utility functions to zip and unzip, detect versions currently installed, check missing files in these versions.

## Function master

**MATLAB init**

Collect information given in settings.ini and save them in some objects of classes specially designed for the recording of Ras variables.

Then, the Ras end of line character is collected because it is different from the newline character of MATLAB.

Finally, a function which makes a loop similar with the main for loop, but without code inside, count the number of iterations necessary, e.g. number of steps of simulation, to perform the entire simulation.

**Loop**

Nb_of_iteration steps will be performed. At each time step, input and output script will be called, when required.

*If it is the first step.*

Then, master instantiates the Ras project, i.e. makes a first simulation step, where a restart file is created at the end of simulation.

*Else if it is the second step*

Then, master updates Ras settings to choose the restart file created during the previous step.

*Else*

Then, it is the third step or more, master updates useful parameters and launch HEC-RAS on a new simulation step, based on the restart file generated at the end of previous step.

**End**

Display results, updates Ras files, to get it ready for a manual simulation without pause.