Scripts input et output

Documentation Interface MATLAB HEC-RAS, IMTLD

Dernière M.A.J : 07/09/2020 13:06:00 – version BETA 1_0_1

Ce document contient toutes les informations nécessaires pour programmer une règle dans l'interface MATLAB HEC-RAS.

Un exemple reprenant la plupart des concepts de cette documentation est la vidéo tutoriel 1 - exemple simple d'automatisation, fournit avec cette documentation. Ce tutoriel explique comment programmer l'ouverture d'une vanne en fonction d'une hauteur d'eau mesurée.

Table des matières

Table des illustrations

Fonctionnalités communes

Variables de la fonction master

Documentation des variables (en anglais)

La documentation détaillée des variables présentées dans la liste au paragraphe suivant, est disponible en écrivant dans la console MATLAB :

```
>> help Results
  Results class def
  to store and save results obtained at each HEC-RAS computation
  because HEC-RAS doesn't save previous results, HEC-RAS overwrite
  previous results

    Documentation for Results
```

Puis en cliquant sur le lien '' Documentation for Results_ ''

Liste des variables disponibles

Conventions :

- Une simulation = une itération = un pas de calcul = un démarrage puis arrêt d'HEC-RAS
- Une date est un objet qui contient toutes les informations caractérisant un instant temporel : du nombre d'années au nombre de secondes
- Une durée est un objet qui contient le nombre de jours, heures, minutes, secondes écoulées entre deux dates

Il est possible d'appeler toutes les variables présentent dans la liste ci-dessous. Pour en savoir plus à propos d'une variable, et connaitre toutes les sous-variables (properties) disponibles. Consulter la documentation des variables, cf. paragraphe précédent.

Tableau 1 - variables du master

syntaxe pour l'appeler	description
sett	settings = réglages ; objet qui contient la liste des réglages choisis par l'utilisateur dans le fichier settings.txt
files	regroupe les noms et chemins des fichiers HEC-RAS
results	Tableau d'objets de type Results_, un objet Results_ par XS
rp	processus RAS, utilisé par certaines fonctions pour interroger HEC-RAS directement
start	date de début du pas de simulation actuel
first	date de début du premier pas de simulation
last	date de fin du dernier pas de simulation
end	date de fin du pas de simulation actuel
time_step	pas de temps, durée d'une simulation
RASnewline	caractère de nouvelle ligne propre aux fichiers HEC-RAS
iteration_nb	le numéro de la simulation actuelle, le premier pas a le numéro 1, et iteration_nb est ensuite incrémenté de 1 à chaque simulation

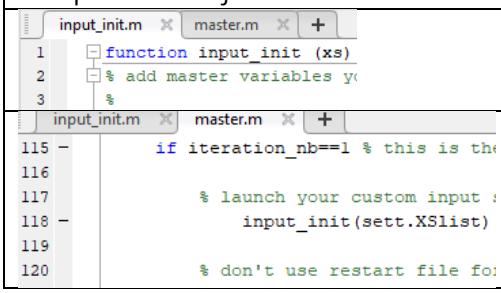
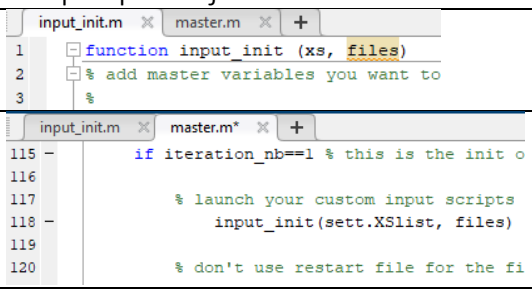
Tableau 2 - sous-variables utiles

syntaxe pour l'appeler	description
sett.XSlist	liste des objets XS choisis dans settings.ini
results{<numéro de XS>}.Array	tableau des résultats pour une XS donnée

Pour appeler une nouvelle variable, il faut ajouter son nom dans les parenthèses de la fonction d'input ou d'output concernée à 2 endroits : dans le script master lors de l'appel de cette fonction, et dans le script d'input et d'output.

Exemple : ajout de la variable files

Voici l'ensemble des modifications à effectuer si vous voulez pouvoir appeler la variable files dans votre script input_init.

Script avant ajout	Script après ajout
	

On remarque que la variable sett.XSlist du master change de nom et est appelée xs dans la fonction input_init. Car rien n'oblige la variable locale à avoir le même nom que la variable du script principal. Cf. cours de programmation ou documentation MATLAB.

Scripts input

Caractéristiques

Les scripts d'input sont :

- input_before_init qui est une fonction appelée seulement avant le premier pas d'initialisation de RAS (lors de ce pas, aucun restart file n'est utilisé car c'est le premier)
- input_before_step qui est une fonction appelée avant tous les pas (sauf le premier)

Structure du code

Etapes que peut suivre votre code :

1. Récupérer ou initialiser variable1 avec une valeur intéressante pour vous (par exemple la hauteur d'eau lors du pas précédent)
2. Règle logique qui change la valeur de variable2 qui sera écrite dans HEC-RAS selon l'état de variable1
3. Sauvegarde dans la input_list avec les paramètres de votre choix pour que MATLAB comprenne bien à quel endroit il doit enregistrer variable2 dans les fichiers d'HEC-RAS

Si vous rencontrez des erreurs, elles peuvent se produire dans des fonctions de bas niveau, i.e. qui ont beaucoup de fonctions intermédiaires. Dans ce cas, cherchez d'abord à vérifier vos arguments avant de tenter de modifier la fonction qui semble poser un problème.

Objet input_list

Cet objet est l'une des principales interfaces entre votre code et le mien. Il est donc très important d'utiliser les paramètres présentés dans le tableau ci-dessous pour éviter tout bug.

Data étant une variable, les valeurs qu'elle peut prendre dépendent de tous les autres paramètres.

Infos supplémentaires est un champ inutilisé pour l'instant. Mais qui pourra l'être si le script prend en charge la modification de nouveaux types de paramètres.

Pour les champs <vide> vous pouvez réduire la taille du tableau si c'est possible ou écrire n'importe quoi dans cette case, son contenu ne sera pas pris en compte.

Dans le tableau 3 des valeurs possibles des principaux paramètres, les paramètres ref_param et xs sont exclus car ils ont trop de valeurs possibles.

Les valeurs possibles de ref_param seront détaillés plus loin, dans le tableau 5.

Il n'est pas pertinent de détailler les valeurs possibles des xs, car cela dépend de chaque projet.

Tableau 3 - valeurs possibles des principaux paramètres

Nom français	Type (pour l'interface)	Extension du fichier RAS	Infos supplémentaires
Nom anglais	Type	Fileext	other_info
Valeurs possibles	hydrograph	.u	<vide>
	param	.u	<vide>
	param	.p	<vide>

Tableau 4 - valeurs possibles de la variable data en fonction du paramètre ''type''

Type	Format de data possible
hydrograph	Cell_array horodaté [1]
hydrograph	Tableau exclusivement numérique
param	Chaine de caractères [2]

Description des types

Hydrograph = hydrogramme

Param = un paramètre indépendant, il ne doit apparaitre qu'une fois dans le fichier cible de Ras.

Si vous souhaitez éditer un paramètre dépendant, i.e. apparaissant plusieurs fois mais distinguable car associé à une XS différente. Je vous invite à consulter la documentation pour développeurs au paragraphe dépendance de paramètres.

Tableau 5 : combinaisons possibles de paramètres

Dans ce tableau, la liste des combinaisons essaie d'être exhaustive, mais il est possible que certaines combinaisons non testées fonctionnent. Si vous en trouvez, je vous invite à les signaler.

[1] Tout une ligne ou tout une colonne contient des objets dates MATLAB ou des chaines de caractère au format 22-Jan-2020 11:00:00, une autre contient uniquement des nombres décimaux.
[2] Cette chaine sera écrite tel quel dans les fichiers HEC-RAS, MATLAB n'opère aucune vérification quant à sa pertinence.

Tableau 5 - Combinaisons possibles de paramètres

Nom français	Type (pour l'interface)	Extension du fichier RAS	Nom du paramètre de référence[3]	Station de mesure où s'applique ce paramètre	Infos supplémentaires
Nom anglais	Type	Fileext	Ref_param	XS	other_info
Valeurs possibles	hydrograph	.u	Flow hydrograph	objet de classe XS, pas sur un bâtiment	<vide>
	hydrograph	.u	Stage hydrograph	objet de classe XS, pas sur un bâtiment	<vide>
	hydrograph	.u	Stage and Flow Hydrograph	objet de classe XS, pas sur un bâtiment	<vide>
	hydrograph	.u	Gate Name=Gate #1	objet de classe XS, **sur** un bâtiment	<vide>
	hydrograph	.u	Uniform Lateral Inflow Hydrograph	objet de classe XS, pas sur un bâtiment	<vide>
	rating curve	.u	Rating Curve	objet de classe XS, seulement sur une extrémité	<vide>
	param	.u	toute chaine de caractère présente une seule fois dans le fichier	<vide>	<vide>
	param	.p	toute chaine de caractère présente une seule fois dans le fichier	<vide>	<vide>

[3] Nom exacte du paramètre tel qu'écrit dans les fichiers de RAS

Scripts output

Caractéristiques

Les scripts d'output sont :

- output_after_step ; fonction qui s'exécute après chaque exécution de RAS
- output_after_all ; fonction qui s'exécute une seule fois, à la toute fin du script

Structure du code

Dans l'exemple de structure de code ci-dessous, il s'agit davantage d'un exemple de structure simple, que d'une structure à reproduire dans tous les cas. Il y a de nombreuses tâches que vous pouvez réaliser après l'exécution de chaque pas.

1. Déclaration des variables globales qui vont être utilisées dans les scripts d'input
2. Eventuellement, traitement des données
3. Initialisation de ces variables avec des valeurs extraites des résultats via l'objet results

Objet results

results est un cell array, c'est-à-dire un tableau pouvant contenir des objets de classe différente dans chaque cellule. Ce tableau contient des objets de classe Results_.

Donc results{1} appelle le contenu de la première case du tableau results, qui est un objet de type Results_. Cet objet n'a pas de nom ''classique'', la seule manière de le dénommer est de l'appeler comme précédemment : results{1}

Pour faciliter la compréhension, chaque case du tableau results contient toutes les valeurs relevées à une XS. Donc results{1} contient toutes les valeurs relevées en XS1 (selon ce que vous y avez mis dans settings.ini).

Cependant, lors de l'appel de la méthode GetValue associée à l'objet results{1}, on précise la xs de même numéro (pour des raisons de programmation).

Objet XS

Les XS sont structurées de la même manière que les Results_.

Pour récupérer la xs de numéro 1 par exemple, il suffit de récupérer le contenu de la première case du tableau des XS qui s'appelle xs.

Donc xs{1} appelle un objet de type XS contenant toutes les informations de position d'une station.

Objet rp

Il s'agit du process RAS.

ATTENTION : RAS doit rester ouvert en arrière-plan lorsque MATLAB écrit dans les paramètres, cela ne pose pas de problème et permet à MATLAB de demander des informations à RAS. Cf. documentation développeur pour plus d'informations.

Ici, les méthodes des objets Results_ ont besoin de communiquer avec RAS. Il faut donc leur donner le nom de leur interlocuteur. C'est le rôle du paramètre rp.

Exemple d'utilisation du script output_after_all

Dans le cadre de l'automatisation d'un système réel : si on simule 24 heures par pas de 10 minutes, cela génère une grande quantité de valeur. Si vous êtes en train de commander un système réel, il peut être pratique de recommencer une nouvelle série de simulation afin de garder un nombre de valeurs à stocker simultanément relativement faible.

Ainsi on peut par exemple à la fin du master appeler un script auto-launcher (non fournit pour l'instant) qui sauvegarde les résultats dans un tableur et relance la simulation pour les 24 heures suivantes.

Dans le cadre d'une simulation qui tourne plusieurs années par exemple. Cette méthode permettrait un gain en mémoire important. Car il éviterait le stockage et le maintien en mémoire vive des données des jours précédents.