



TP5 : Utilisation de données d'entrées

Objectifs: Utiliser des données en entrée en tenant compte des conditions requises/utilisées

Pré-requis: TP3,TP4

Nous allons (encore) améliorer le simulateur (`training.su.prod`), créé lors du TP3, en ajoutant le coefficient de rétention S comme donnée d'entrée distribuée du simulateur. Ceci permet de valuer ce coefficient de manière indépendante pour chaque SU. Une fois cette modification apportée, le simulateur pourra utiliser, par ordre de priorité, la donnée d'entrée distribuée, le paramètre de simulateur ou la valeur par défaut.

1 Code source

Les modifications du code source vont porter sur la signature et sur la méthode `runStep()`.

1.1 Signature

Nous allons déclarer la prise en compte d'une donnée d'entrée distribuée facultatif pour le coefficient de rétention (nommé s). Cette déclaration se fait au travers de l'instruction `DECLARE_USED_INPUTDATA`.

Une fois complétée, la signature devrait être similaire à :

```
BEGIN_SIMULATOR_SIGNATURE("training.su.prod")

DECLARE_NAME("");
DECLARE_DESCRIPTION("");

DECLARE_VERSION("13.05");
DECLARE_STATUS(openfluid::ware::EXPERIMENTAL);

DECLARE_DOMAIN("");
DECLARE_PROCESS("");
DECLARE_METHOD("");
DECLARE_AUTHOR("", "");
```

```

DECLARE_SIMULATOR_PARAM("s", "", "-");
DECLARE_REQUIRED_VAR("water.atm-surf.H.rain", "SU", "rainfall_height_on_the_SU", "m");

DECLARE_PRODUCED_VAR("water.surf.H.runoff", "SU", "water_runoff_height_on_surface_of_SU", "m");
DECLARE_PRODUCED_VAR("water.surf.H.infiltration", "SU",
    "water_infiltration_height_through_the_surface_of_SU", "m");

// declaration de la donnee d'entree facultative nommee s
DECLARE_USED_INPUTDATA("s", "SU", "", "-");

// Scheduling
DECLARE_SCHEDULING_DEFAULT;

END_SIMULATOR_SIGNATURE

```

1.2 runStep()

La modification à apporter dans le `runStep()` concerne la prise en compte de la donnée d'entrée `S` uniquement s'il existe. La présence de cette donnée d'entrée peut être testée au travers de l'instruction `OPENFLUID_IsInputDataExist`. La valeur de la donnée d'entrée peut être récupérée au travers de l'instruction `OPENFLUID_GetInputData`.

Une fois complétée, la méthode `runStep()` devrait être similaire à :

```

openfluid::base::SchedulingRequest runStep()
{
    openfluid::core::Unit* pSU;
    openfluid::core::DoubleValue RainValue;
    openfluid::core::DoubleValue RunoffValue;
    openfluid::core::DoubleValue InfiltrationValue;
    double S;

    OPENFLUID_UNITS_ORDERED_LOOP("SU", pSU)
    {
        S = m_S; // Coeff. de retention recoit la valeur par default

        // recuperation de la valeur du signal de pluie
        OPENFLUID_GetVariable(pSU, "water.atm-surf.H.rain", RainValue);

        // recuperation du s par SU s'il existe
        if (OPENFLUID_IsInputDataExist(pSU, "s")) OPENFLUID_GetInputData(pSU, "s", S);

        // calcul du ruissellement selon la methode SCS
        RunoffValue = 0.0;
        if (RainValue > (0.2*S))
        {
            RunoffValue = std::pow(RainValue - (0.2*S), 2) / (RainValue + (0.8*S));
        }
        // calcul de l'infiltration par deduction
        InfiltrationValue = RainValue - RunoffValue;

        // production de l'infiltration et du ruissellement
        OPENFLUID_AppendVariable(pSU, "water.surf.H.infiltration", InfiltrationValue);
        OPENFLUID_AppendVariable(pSU, "water.surf.H.runoff", RunoffValue);
    }
    return DefaultDeltaT();
}

```

2 Simulation

Dans le jeu de données d'entrée, la donnée d'entrée distribuée `s` doit être présent pour chaque SU. Pour cela, il faut modifier le fichier `SU.ddata.fluidx` du jeu de données fourni.

2.1 ... avec l'interface OpenFLUID-Builder

Afin de repartir du TP précédent, créer un projet OpenFLUID nommé TP5 et y importer le jeu de données d'entrée du TP4.

Dans l'onglet `Domaine spatial`, double-cliquer sur SU (15 unités), dans l'onglet `Données spatiales [SU]/inputdata`, cliquez sur l'icône + et rajouter un `inputdata` nommé `s` à la classe d'unité SU, et affecter une valeur comprise entre 0.0001 et 0.008 à chaque unité.

Lancer la simulation.

2.2 ... en ligne de commande

Une fois complété, le fichier `SU.ddata.fluidx` devrait être structuré comme suit :

```
<?xml version="1.0" standalone="yes"?>
<openfluid>
  <domain>
    <inputdata unitclass="SU" colorder="area;slope;flowdist;s" >
1 5427.693909 0.02172 41.6 0.006
2 16529.924492 0.0001 49.2 0.002
3 3085.65168 0.01789 26.7 0.004
4 3556.816467 0.0192 51.4 0.007
5 5167.890732 0.02631 52.3 0.004
6 7144.493736 0.11577 85.4 0.0001
7 7291.455818 0.16371 38.6 0.0007
8 8656.038666 0.04661 41 0.006
9 8867.89624 0.11799 80.6 0.004
10 2559.261559 0.20639 44.1 0.007
11 10598.565178 0.05499 43.4 0.006
12 2558.575302 0.0651 31.1 0.004
13 3335.563385 0.01797 51.7 0.001
14 2235.471466 0.02281 21.7 0.004
15 5231.501724 0.06828 40.5 0.003
    </inputdata>
  </domain>
</openfluid>
```

La commande à exécuter est donc :

```
openfluid -i /home/openfluid/formation/datasets/TP1-TP7
-o /home/openfluid/formation/outputs/TP5
```

(à taper sur une seule ligne)

Si tout s'est bien passé, les résultats de la simulation sont accessibles dans `/home/openfluid/formation/outputs/TP5`.