OpenFLUID
Software Environment
for Modelling Fluxes in Landscapes

Tutoriel : Utilisation du package ROpenFLUID et illustration par une analyse de sensibilité de certains paramètres du modèle MHYDAS

Objectifs:	Connaître les principales commandes du package ROpenFLUID et avoir un aperçu des potentialités de R sur l'analyse de modèle
Pré-requis:	TP0 - TP MHYDAS - notions en langage R

1 Introduction

Le package ROpenFLUID permet l'interaction entre l'environnement OpenFLUID et le langage de programmation R. Il permet notamment de lancer une simulation OpenFLUID dans l'environnement R, définir, obtenir ou modifier l'ensemble des paramètres d'une simulation (temps caractéristiques, données spatialisées, paramètres de simulateur) et de récupérer les résultats de la simulation sous forme de vecteurs R. Son installation et son utilisation sont décrites sur le site web OpenFLUID Community : http ://www.openfluid-project.org/community/ à la rubrique : "Using OpenFLUID within the GNU R environment".

Ce tutoriel présente tout d'abord les commandes permettant de lancer une simulation et récupérer ses résultats. Nous utiliserons ensuite les fonctionnalités du package R sensitivity pour analyser une version simplifiée du modèle MHYDAS.

2 La simulation exemple

Nous utilliserons ici le modèle MHYDAS sur un domaine spatial restreint. Le domaine spatial est composé de 6 parcelles (SU) et des 3 tronçons de fossés (RS). Il est décrit par la figure suivante

Le modèle utilisé est composé

- d'un simulateur produisant la pluie sur les parcelles
- d'un simulateur partageant infiltration et ruissellement selon la méthode de Morel-Seytoux
- d'un simulateur transférant le ruissellement sur les parcelles selon le modèle de l'onde diffusante (résolution d'Hayami)
- d'un simulateur transférant le ruissellement dans le réseau hydrographique selon le modèle de l'onde diffusante (résolution d'Hayami)

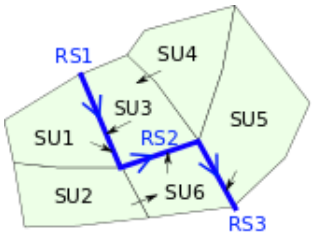

Figure 1 – bassin versant virtuel considéré par le modèle MHYDAS dans ce tutoriel

3 Quelques opérations basiques

3.1 Le lancement d'une simulation

Le lancement d'une simulation dans un environnement R est effectué de la manière suivante :

- chargement du package ROpenFLUID dans l'environnement R
- définition de la simulation OpenFLUID grâce à son jeu de données d'entrée
- définition du répertoire des fichiers résultats
- définition d'un format des fichiers résultats adapté à R
- lancement de la simulation

Le script R correspondant à ces opérations peut être celui-ci :

```
library("ROpenFLUID")     # should be done only once

# definition of the paths and OpenFLUID simulation
OF.dirBase = "/home/user/OpenFLUID-Projects/TP_ROpenFLUID/" # directory of OpenFLUID
    project
OF.dirIn   = paste(OF.dirBase,"IN",sep="")
OF.dirOut  = paste(OF.dirBase,"OUT",sep="")
OF.simu    = OpenFLUID.openDataset(OF.dirIn)  # OpenFLUID simulation
OpenFLUID.setCurrentOutputDir(OF.dirOut)      # directory of output files

# test of OpenFLUID simulation
OpenFLUID.addVariablesExportAsCSV(OF.simu,'RS')
OpenFLUID.addVariablesExportAsCSV(OF.simu,'SU')
OpenFLUID.runSimulation(OF.simu)
```

Note: Une fois le package ROpenFLUID chargé dans l'environnement R sous la console, il est possible de visualiser l'ensemble des commandes du package en entrant à l'invite de commande > `OpenFLUID.` suivi de deux fois la touche TAB.

Note: De même, il est possible de connaître rapidement les arguments d'une commande en particulier (`OpenFLUID.runSimulation` par exemple) en entrant à l'invite de commande > `OpenFLUID.runSimulation(` suivi de deux fois la touche TAB.

3.2 Une exploitation simple des résultats de la simulation

Une fois la simulation lancée grâce au script précédent, les résultats sont enregistrés comme fichiers sur le disque avec un format spécifique. Le script suivant permet d'enregistrer comme

objet R la pluie, le ruissellement et l'infiltration de la SU5, ainsi que le débit et la hauteur d'eau de la RS3 (l'exutoire du domaine).

```
# loading of some results
OF.H.rain           = OpenFLUID.loadResult(OF.simu,"SU",5,"water.atm-surf.H.rain")
OF.H.runoff         = OpenFLUID.loadResult(OF.simu,"SU",5,"water.surf.H.runoff")
OF.H.infiltration   = OpenFLUID.loadResult(OF.simu,"SU",5,"water.surf.H.infiltration")
OF.Q.downstream     = OpenFLUID.loadResult(OF.simu,"RS",3,"water.surf.Q.downstream-rs")
OF.H.level          = OpenFLUID.loadResult(OF.simu,"RS",3,"water.surf.H.level-rs")
```

Ces vecteurs peuvent désormais être exploités sous R. Les scripts ci-dessous permettent de :
- calculer le débit cumulé et la hauteur eau maximale à l'exutoire du bassin versant
- tracer un graphique représentant la pluie, le ruissellement et l'infiltration sur la SU5

```
# screen output of some results
OF.deltaT  = OpenFLUID.getDeltaT(OF.simu)     # time step duration
print(paste("cumulative flow at the outlet of the catchment :
",as.character(sum(OF.Q.downstream[,2])*OF.deltaT)," m3",sep=""))
print(paste("maximum water level at the outlet of the catchment :
",as.character(max(OF.H.level[,2]))," m",sep=""))
```

```
# plot of some results
plot(OF.H.rain[,1],OF.H.rain[,2],type='l',lwd=3,xlab='Date',ylab='Flow [m3.s-1]')
lines(OF.H.infiltration[,1],OF.H.infiltration[,2],type='l',col='red',lwd=2)
lines(OF.H.runoff[,1],OF.H.runoff[,2],type='l',col='blue',lwd=2)
legend("topright",c('Rain on SU5','Runoff on SU5','Infiltration on SU5'),lty=c(1,1,1),
    lwd=c(3,2,2),col=c('black','red','blue'))
```

débit cumulé à l'exutoire	1.0411 m^3
hateur d'eau maximale à l'exutoire	0.0109 m

Table 1 – Résultats globaux de la simulation

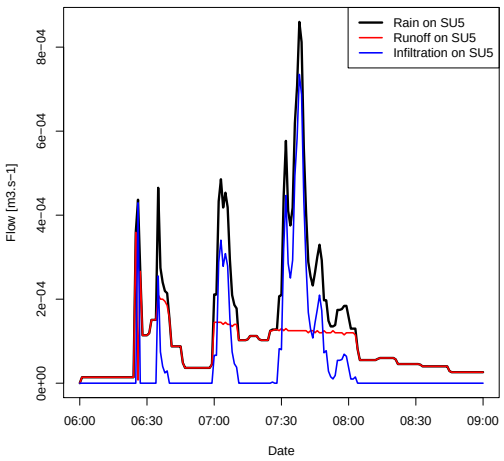

Figure 2 – graphique représentant la pluie, le ruissellement et l'infiltration de la SU5 sur la période de simulation

4 L'analyse statistique d'un modèle

Les opérations précédentes illustrent une utilisation pratique, mais basique de l'environnement R. Elles peuvent cependant être réalisées simplement avec d'autres outils en dehors de l'environnement R.

Un des intérêts de lancer des simulations OpenFLUID dans l'environnement R est de pouvoir réaliser simplement des multi-simulations et d'exploiter leurs résultats en bénéficiant de toutes les fonctionnalités de R. Dans la suite de ce tutoriel, nous analyserons 3 paramètres du modèle :

- C_{RS} : la célérité moyenne sur les RS (paramètre du simulateur transferts sur les fossés)
- $\theta_{I\ SU5}$: l'humidité initiale de la parcelle 5 (donnée spatialisée de la RS3)
- L_{RS3} : la largeur du fossé 3 (donnée spatialisée de la RS3)

sur 2 de ces sorties :

- Q_{RS3} : le débit cumulé à l'exutoire
- H_{RS3} : la hauteur d'eau maximale à l'exutoire

4.1 Définition d'une fonction R permettant des multi-simulations

Dans l'exemple choisi dans ce tutoriel, les multi-simulations sont lancées grâce à une seule fonction prenant en argument une matrice où chaque ligne correspond à un jeu de données à analyser $X = (C_{RS}, \theta_{I\ SU5}, L_{RS3})$ et un tag `varOut` "H" ou "Q". La fonction retourne un vecteur où chaque élément correspond au résultat de chaque simulation : la hauteur d'eau maximale de la RS3 si le tag est "H", son débit cumulé si le tag est "Q".

```
# definition of the function running OpenFLUID with the factors set X
OF.multiRun <- function(X,varOut) {

  out = rep(NA,nrow(X))
```

```
  # loop on the lines of the factors vector
  for (i in seq(1:nrow(X))) {

    # redefine simulator parameters and input data according to vector X
    OpenFLUID.setSimulatorParam(OF.simu,"water.surf.transfer-rs.hayami","meancel",X[i
        ,1])
    OpenFLUID.setInputData(OF.simu,"SU",5,"thetaini",X[i,2])
    OpenFLUID.setInputData(OF.simu,"RS",3,"width",X[i,3])

    # run of OpenFLUID with new simulator parameters and input data
    OpenFLUID.addVariablesExportAsCSV(OF.simu,'RS')
    OpenFLUID.addVariablesExportAsCSV(OF.simu,'SU')
    OpenFLUID.runSimulation(OF.simu)

    # choice of output variables
    if (varOut=="Q") {
      outOF  = OpenFLUID.loadResult(OF.simu,"RS",3,"water.surf.Q.downstream-rs")
      out[i] = sum(outOF[,2])*OpenFLUID.getDeltaT(OF.simu)
    }
    else if (varOut=="H") {
      outOF  = OpenFLUID.loadResult(OF.simu,"RS",3,"water.surf.H.level-rs")
      out[i] = max(outOF[,2])
    }
  }
  return(out)
}
```

Deux nouvelles commandes du package ROpenFLUID sont introduites dans ce script : `OpenFLUID.setSimulatorParam` et `OpenFLUID.setInputData` qui permettent de modifier respectivement un paramètre de simulateur et une donnée d'entrée spatialisée.

4.2 Propagation d'incertitudes et intervalle de confiance des résultats

La simulation utilisée dans ce tutoriel demande en entrée des paramètres et des données d'entrée. Ces facteurs peuvent être issus de mesures, de calculs, définis par dire d'experts, etc...En connaissant l'incertitude sur ces facteurs, nous pouvons analyser la propagation de ces incertitudes sur les résultats du modèle et en déduire leur intervalle de confiance. Cette étude sera réalisée grâce à une analyse statistique des sorties d'un grand nombre de simulation.

La première phase de cette étude est de définir l'incertitude des facteurs à analyser. Nous définissons une distribution uniforme des 3 facteurs étudiés autour de la valeur connue et lançons les simulations.

```
# test of the multi-simulation function

# definition of the analysed parameters and their characteristics
OF.caractFactor = list(
  name = c("Mean celerity on RS","Initial moisture on SU5","Width of RS3"),
  ref  = c(0.6,0.3 ,0.50),
  inf  = c(0.2,0.25,0.35),
  sup  = c(1.0 ,0.35,0.65)
)

# definition of the factors set
N = 1000
v = apply(matrix(runif(3*N),nrow=N,ncol=3),
```

```
          MARGIN=1, function(x) {OF.caractFactor$inf + x*(OF.caractFactor$sup-OF.
              caractFactor$inf)})
v = t(v)
colnames(v) <- OF.caractFactor$name

# launch of the multi-simulation
waterLevel = OF.multiRun(v,"H")
```

L'environnement R permet par exemple de visualiser la distribution des résultats des simulations. Il permet également d'évaluer la hauteur d'eau moyenne sur l'ensemble des résultats, la médiane des résultats et l'intervalle de confiance à 90%, c'est à dire, l'intervalle sur lequel le résultats a une probabilité de plus de 90% d'être exact (si on ne considère que l'incertitude sur les facteurs comme source d'erreur).

```
# statistical uncertainty analysis
hist(waterLevel)
mean(waterLevel)
quantile(waterLevel, probs = c(0.05, 0.5, 0.95))
```

La même étude a été menée en modifiant l'incertitude sur les facteurs. Le tableau suivant rassemble les caractéristiques des distributions étudiées et les résultats associés.

Facteur analysé	C_{RS}	$\theta_{I\ SU5}$	L_{RS3}
distribution 1	$0.6 \pm 67\%$	$0.3 \pm 16\%$	$0.5 \pm 30\%$
distribution 2	$0.6 \pm 16\%$	$0.3 \pm 16\%$	$0.5 \pm 30\%$
distribution 3	$0.6 \pm 67\%$	$0.3 \pm 16\%$	$0.5 \pm 10\%$

Table 2 – Caractéristiques des incertitudes testées pour les 3 facteurs considérés

	médiane $[m]$	intervalle de confiance à 90% $[m]$	taux d'incertitude
distribution 1	0.0107	de 0.00912 à 0.01296	$\pm$ 17%
distribution 2	0.0110	de 0.00935 à 0.01311	$\pm$ 16%
distribution 3	0.0108	de 0.01019 à 0.01145	$\pm$ 6%

Table 3 – Propagation d'incertitudes sur les résultats des simulations pour les 3 distributions de facteurs considérés

Cette étude permet d'associer une marge d'erreur au résultat de la simulation en fonction des incertitudes des paramètres et données spatialisées.

L'étude des médianes, intervalles de confiance et distributions des hauteurs d'eau pour les 3 distributions de facteurs permet de tirer les informations suivantes :

- une faible dérive de la médiane par rapport à la valeur calculée au paragraphe 3.2,
- l'assymétrie de la distribution des hauteurs d'eau,
- une incertitude des résultats moins importante que la plus grande des incertitudes sur les paramètres,

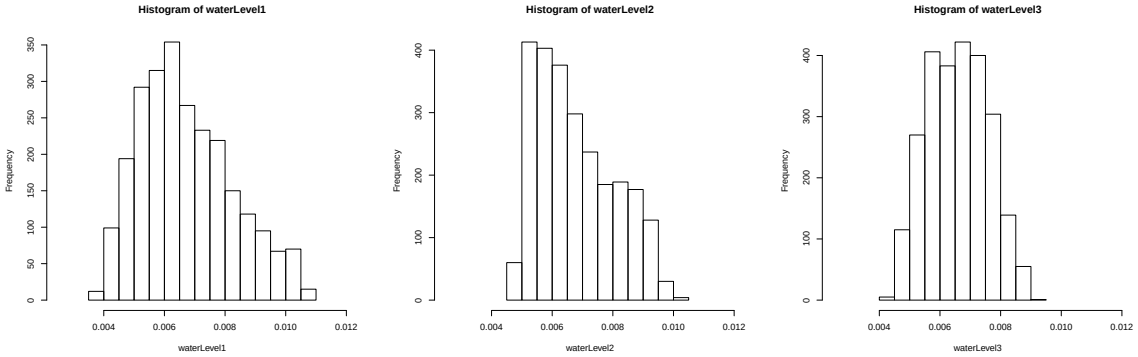

Figure 3 – distributions des hauteurs d'eau simulées par MHYDAS en considérant une distribution d'incertitude uniforme pour les facteurs C_{RS},$\theta_{I\ SU5}$ etL_{RS3} - les 3 graphiques montrent l'effet d'une modification des incertitudes sur les facteurs

- une amélioration de la précision des résultats en réduisant l'incertitude des paramètres, mais très relative selon les paramètres.

Ces informations sont cependant valables uniquement dans cette situation précise : le même modèle, les mêmes valeurs et incertitudes des paramètres, les mêmes sorties analysées.

Note: Nous n'avons considéré dans cet exemple que l'incertitude sur les données d'entrée. Rappelons que d'autres sources d'erreur existent : hypothèses des modèles, approximation des équations, arrondies numériques,...

4.3 Analyse de sensibilité d'un modèle

L'analyse précédente a permis d'évaluer la manière dont se propageait les incertitudes sur les paramètres d'un modèle. Lorsque l'erreur sur les sorties est trop importantes, il est parfois possible de faire un effort sur quelques paramètres pour réduire leur incertitude. Il est souvent très difficile de faire cet effort sur l'ensemble des facteurs. De même, lors du calage d'un modèle (trouver le jeu de facteurs optimal pour retrouver des sorties mesurées), il est intéressant de connaître la sensibilité des paramètres sur les sorties du modèle pour savoir sur quelques paramètres faire porter l'effort de calage. Le package `sensitivity` de R permet ces études. Le script suivant permet cette analyse avec deux méthodes (Morris et Fast99) parmi celles mis à disposition dans le package.

```
# load of sensitivity package
library("sensitivity")   # should be done only once

# Morris sensitivity analysis
OF.morris.Q <- morris(model=OF.multiRun,
                      factors=as.character(OF.caractFactor$name),r=500,
                      design=list(type="oat", levels=25, grid.jump=12),
                      binf=c(0.1,0.05,0.1),
                      bsup=c(1.0,0.35,1.0),
                      varOut="Q")
# Morris sensitivity analysis
OF.morris.H <- morris(model=OF.multiRun,
                      factors=as.character(OF.caractFactor$name),r=500,
                      design=list(type="oat", levels=25, grid.jump=12),
                      binf=c(0.1,0.05,0.1),
                      bsup=c(1.0,0.35,1.0),
                      varOut="H")

plot(OF.morris.Q,xlim=c(0,1),ylim=c(0,1))
title(main="Morris sensitivity analysis on the downstream")
plot(OF.morris.H,xlim=c(0,1),ylim=c(0,1))
title(main="Morris sensitivity analysis on the water level")

# FAST sensitivity analysis
OF.bounds <- apply(cbind((c(0.1,0.05,0.1)), (c(1.0,0.35,1.0))),MARGIN=1,function(x){
    list(min=x[1],max=x[2])} )
print(OF.caractFactor$inf)
print(OF.caractFactor$sup)
print(OF.bounds)
OF.fast99.Q <- fast99(model=OF.multiRun,
                      factors=as.character(OF.caractFactor$name),
                      n=500,
                      q="qunif",
                      q.arg=OF.bounds,
                      varOut="Q")
OF.fast99.H <- fast99(model=OF.multiRun,
                      factors=as.character(OF.caractFactor$name),
                      n=500,
                      q="qunif",
                      q.arg=OF.bounds,
                      varOut="H")

plot(OF.fast99.Q)
title(main="FAST sensitivity analysis on the downstream")
plot(OF.fast99.H)
title(main="FAST sensitivity analysis on the water level")
```

Ces méthodes permettent de tracer les figures suivantes :

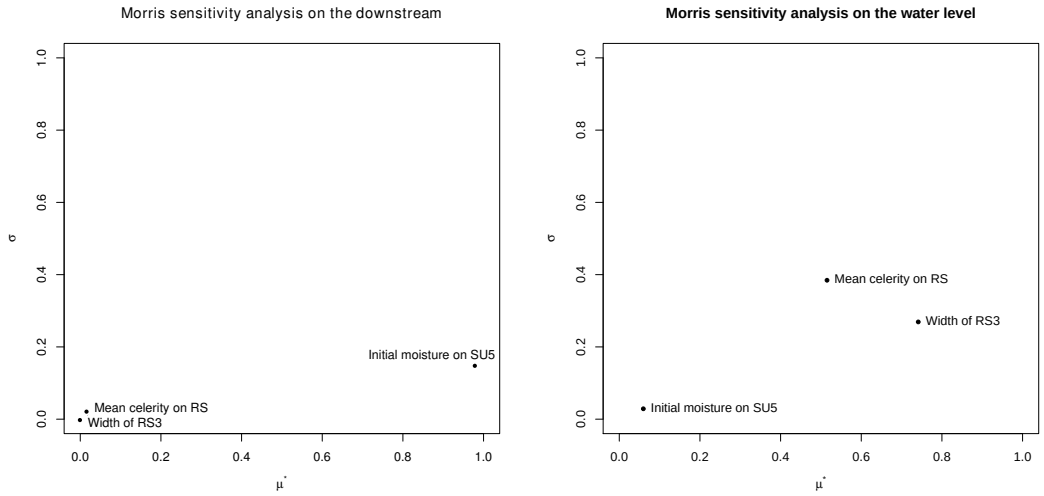

Figure 4 – Analyse de Morris : les points du graphique représentent la variance en fonction de la moyenne des effets élémentaires de n répétitions du modèle (la variation de la sortie lorsqu'on fait varier le paramètre considéré).

Les deux analyses peuvent être interprétées de la même manière. L'humidité initiale de la parcelle 5 est le principal facteur à influencer le débit cumulé à l'exutoire. En effet, la parcelle 5 alimente directement le fossé 3 (voir représentation du bassin versant virtuel fig. 1). Si la parcelle est très humide, le ruissellement est immédiat, au contraire, lorsque la parcelle est sèche, la pluie doit être suffisante pour saturer la parcelle avant qu'il y ait ruissellement. Les autres facteurs ont bien moins d'infuence : la célérité moyenne est un paramètre de calage et la largeur du fossé n'entre pas dans le calcul du débit à l'exutoire. Le calcul de la hauteur d'eau est effectué directement à partir du débit dans le fossé grâce à une courbe de tarage : la largeur du fossé est ici un facteur déterminant sur cette sortie puisqu'elle est utilisée pour définir la courbe de tarage.

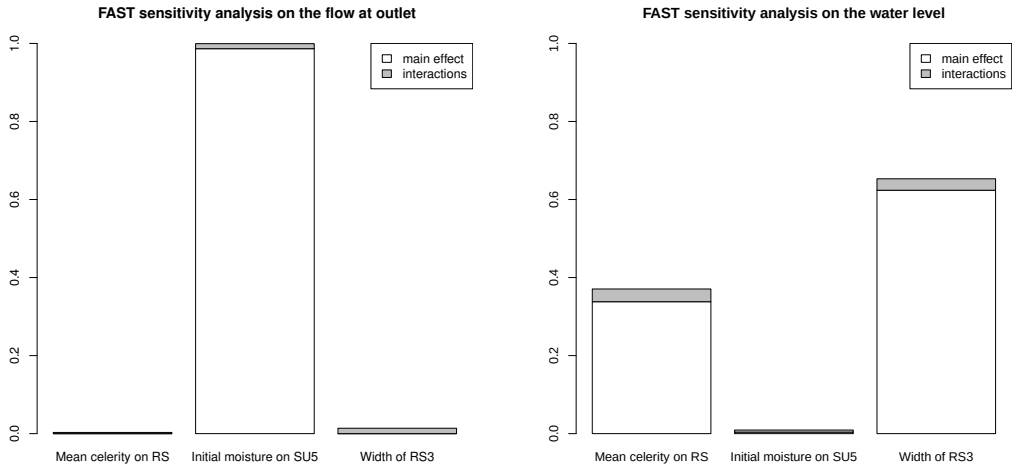

Figure 5 – Analyse Fast99 : les colonnes représentent, en histogramme cumulé, l'effet principal (impact direct et linéaire) du paramètre sur la sortie et ses effets par interactions ou non-linéaire.

Pour illustrer concrêtement cette analyse, 3 simulations ont été lancées : un jeu de données référence, un jeu de données avec une variation de $\theta_{I\ SU5}$ et un jeu de données avec une variation de L_{RS3}. Le tableau suivant montre les impacts sur les deux variables de sortie.

Variable de sortie	**jeu de données de référence**	**variation de $\theta_{I\ SU5}$** (-25 %)	**variation de L_{RS3}** (-25%)
variation de Q	0%	-12%	0%
variation de H	0%	-6%	+34%

Table 4 – Résultats de simulation pour 3 jeux de données. Ces tests confirment l'analyse effectuée précédemment

Note: On note ici que l'influence des paramètres des modèles peuvent être très différents selon la variable de sortie que l'on cherche à analyser

5 Optimisation de modèle

L'étude précédente a permis d'évaluer la sensibilité des paramètres du modèle. Nous allons maintenant chercher à optimiser les deux paramètres les plus sensibles du modèle afin d'obtenir des mesures de hauteur d'eau effectuées à l'exutoire. La langage R dispose d'algorithmes génériques pour optimiser une fonction R. Nous utiliserons ici une méthode quasi-Newton fournie dans l'installation de base de R.

De la même manière que la fonction `multiRun`, nous définissons tout d'abord une fonction qui paramètre la simulation, la lance et calcule l'erreur quadratique moyenne entre les mesures et le résultat simulé.

```
# model optimization
# definition of the function running OpenFLUID with width of RS3 mean celerity as
    unknown variables
OF.singleRun <- function(X,ref) {

    # redefine simulator parameters and input data according to vector X
    OpenFLUID.setSimulatorParam(OF.simu,"water.surf.transfer-rs.hayami","meancel",X[1])
    OpenFLUID.setInputData(OF.simu,"RS",3,"width",X[2])

    # run of OpenFLUID with new simulator parameters and input data
    OpenFLUID.addVariablesExportAsCSV(OF.simu,'RS')
    OpenFLUID.runSimulation(OF.simu)

    # choice of output variables
    outOF = OpenFLUID.loadResult(OF.simu,"RS",3,"water.surf.H.level-rs")
    out   = sqrt(sum((outOF[,2]-ref[,2])**2))
    print(paste("simulation :",X[1],X[2],"RMSE",out,sep=" "))
    return(out)
}
```

Le fichier des hauteurs mesurées est lu et enregistré comme objet R.

```
# reference output definition
OF.refOptim=OpenFLUID.loadResultFile(paste(OF.dirBase,"measured-level.csv",sep=""))
plot(OF.refOptim,xlab='Date',ylab='Water level [m]')
```

L'algorithme d'optimisation est lancé avec un jeu de paramètres initiaux donné par l'utilisateur. Les paramètres optimisés sont indiqués dans le tableau suivant.

```
# run optimization process
OF.parInit = c(0.01,0.2)
OF.factorsOptim=optim(par=OF.parInit,fn=OF.singleRun,method='L-BFGS-B',lower=0.1*OF.
    parInit,upper=5.0*OF.parInit,ref=OF.refOptim)
```

Paramètres	C_{RS}	L_{RS3}
initiaux	0.01	0.2
optimisés	0.05	0.489

Table 5 – Paramètres initiaux utilisés pour l'algorithme `optim` par la méthode `L-BFGS-B` et résultats de l'opimisation

Enfin, une comparaison entre la hauteur d'eau mesurée, la hauteur simulée à partir des paramètres initiaux et des paramètres optimisés peut être tracé.

```
# plot of water level with the initial parameters
OpenFLUID.setSimulatorParam(OF.simu,"water.surf.transfer-rs.hayami","meancel",OF.
    parInit[1])
OpenFLUID.setInputData(OF.simu,"RS",3,"width",OF.parInit[2])
OpenFLUID.addVariablesExportAsCSV(OF.simu,'RS')
OpenFLUID.runSimulation(OF.simu)
```

```
plot(OpenFLUID.loadResult(OF.simu,"RS",3,"water.surf.H.level-rs"),type='l',lwd=2,xlab='
    Date',ylab='Water level [m]')

# plot of water level with the optimized parameters
OpenFLUID.setSimulatorParam(OF.simu,"water.surf.transfer-rs.hayami","meancel",OF.
    factorsOptim$par[1])
OpenFLUID.setInputData(OF.simu,"RS",3,"width",OF.factorsOptim$par[2])
OpenFLUID.addVariablesExportAsCSV(OF.simu,'RS')
OpenFLUID.runSimulation(OF.simu)
lines(OpenFLUID.loadResult(OF.simu,"RS",3,"water.surf.H.level-rs"),col='red',lwd=2)

# comparison with measured water level
points(OF.refOptim,col='blue')
legend("topright",c('Initial parameters','Optimized parameters','Measured'),lty=c
    (1,1,3),lwd=c(2,2,2),col=c('black','red','blue'))
print(OF.factorsOptim1$par)
```

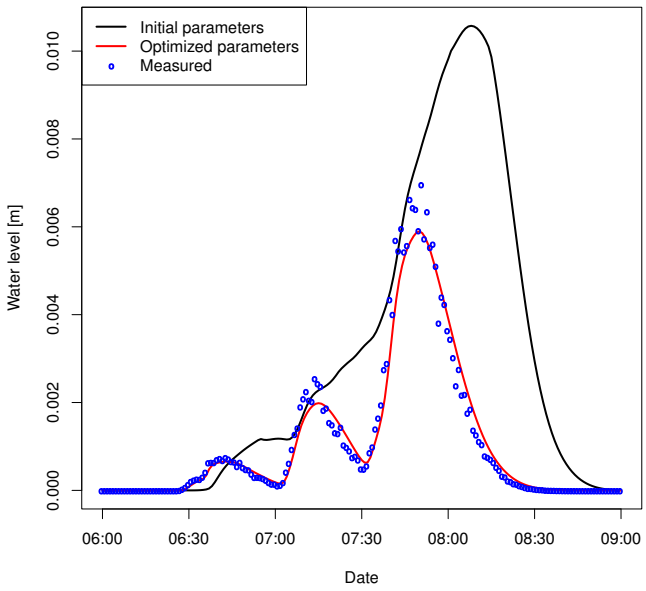

Figure 6 – Évolution de la hauteur à l'exutoire mesurée, simulée avec des paramètres initiaux et avec les paramètres optimisés.

L'algorithme utilisé semble efficace pour déterminer quels paramètres du modèle reproduisent avec une erreur minimale les mesures effectuées. Des packages R peuvent fournir d'autres types d'algorithmes, qu'ils soient génériques ou adaptés pour une situation particulière.

A Liste des commandes du package ROpenFLUID

OpenFLUID.addExtraSimulatorsPaths(paths)
OpenFLUID.addVariablesExportAsCSV(ofblob,unitclass)
OpenFLUID.createInputData(ofblob,unitclass,idataname,idataval)
OpenFLUID.getDeltaT(ofblob)
OpenFLUID.getExtraSimulatorsPaths()
OpenFLUID.getGeneratorParam(ofblob,unitclass,varname,paramname)
OpenFLUID.getInputData(ofblob,unitclass,unitid,idataname)
OpenFLUID.getModelGlobalParam(ofblob,paramname)
OpenFLUID.getPeriodBeginDate(ofblob)
OpenFLUID.getPeriodEndDate(ofblob)
OpenFLUID.getSimulatorParam(ofblob,simid,paramname)
OpenFLUID.getUnitsClasses(ofblob)
OpenFLUID.getUnitsIDs(ofblob,unitclass)
OpenFLUID.getVersion()
OpenFLUID.loadResult(ofblob,unitclass,unitid,varname)
OpenFLUID.loadResultFile(filepath)
OpenFLUID.openDataset(path)
OpenFLUID.openProject(path)
OpenFLUID.printSimulationInfo(ofblob)
OpenFLUID.runProject(path)
OpenFLUID.runSimulation(ofblob)
OpenFLUID.setCurrentOutputDir(path)
OpenFLUID.setDeltaT(ofblob,deltat)
OpenFLUID.setGeneratorParam(ofblob,unitclass,varname,paramname,paramval)
OpenFLUID.setInputData(ofblob,unitclass,unitid,idataname,idataval)
OpenFLUID.setModelGlobalParam(ofblob,paramname,paramval)
OpenFLUID.setPeriodBeginDate(ofblob,begindate)
OpenFLUID.setPeriodEndDate(ofblob,enddate)
OpenFLUID.setSimulatorParam(ofblob,simid,paramname,paramval)