BaM !

Bayesian Model fitting

09/10/2015

# Introduction

BaM est une généralisation de BaRatin des points de vue suivants:

* BaM ignore tout concept de courbe de tarage – BaM cale un modèle quel qu’il soit ;
* Le modèle peut posséder un nombre arbitraire de variables d’entrée / sortie (alors que BaRatin était limité à 1 entrée – 1 sortie) .

Cette note décrit le cadre théorique de BaM ainsi que son utilisation pratique via des fichiers de configuration. Il y a de nombreuses ressemblances avec BaRatin, mais également quelques différences importantes qui seront pointées dans ce document.

# Cadre théorique

On considère un modèle formalisé de la façon suivante :

|  |  |
| --- | --- |
|  | (1) |

*  est une matrice de taille  qui représente les  variables d’entrée du modèle ;
*  est le vecteur de paramètres du modèle, de taille  ;
*  est une matrice de taille  qui représente les  variables de sortie simulées par le modèle.

Remarque : le modèle est écrit sous forme matricielle pour pouvoir gérer les modèles dynamiques (typiquement, un modèle hydrologique). En effet :

* Pour un modèle non-dynamique comme une courbe de tarage, la simulation de la *ième* sortie (le *ième* débit) ne dépend que de la *ième* entrée (la *ième* hauteur), on pourrait donc écrire le modèle sous forme vectorielle  ;
* Ce n’est pas le cas pour un modèle dynamique : la simulation de la *ième* sortie dépend de tout l’historique des entrées, depuis  jusqu’à  (cette dépendance étant introduite par les variables d’état).

Pour le calage du modèle, on dispose d’observations pour les variables d’entrée/sortie, respectivement notées  et .

On utilise le modèle statistique suivant :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | (a) | |  | (b) | |  | (c) | |  | (d) | | (2) |

Les paramètres inconnus à estimer sont:

* Les paramètres du modèle ;
* Les paramètres gouvernant l’écart-type des erreurs structurelles ;
* Les erreurs d’entrée .

Après quelques calculs non détaillés ici, la distribution a posteriori de ces paramètres inconnus s’écrit :

|  |  |
| --- | --- |
| avec  et  la densité d’une loi normale de moyenne *m* et d’écart-type *s*, évaluée en *z*. | (3) |

Remarques

1. La gestion des erreurs d’entrée est très couteuse en termes de complexité, car chaque valeur d’entrée considérée comme incertaine ajoute un paramètre inconnu à estimer ! En pratique on peut spécifier  pour éviter d’introduire un paramètre supplémentaire. Le cas  est extrêmement expérimental pour le moment (il est implémenté mais n’a pas été testé proprement).
2. A noter que les hypothèses de normalité et surtout d’indépendance des erreurs sont omniprésentes. Cette indépendance est à la fois « temporelle » et «inter-variable ». Il est possible de s’affranchir de ces hypothèses, mais je n’ai rien implémenté en ce sens car :
   1. Cela demanderait à l’utilisateur de spécifier des matrices de covariance plutôt que des écart-types (ce qui entraînerait des fichiers de configuration beaucoup plus complexes…)
   2. Une implémentation naïve conduirait à un code très lent car il faudrait sans arrêt inverser des matrices de covariance, mais une implémentation non-naïve n’est pas triviale du tout !

# Fichiers de configuration

Les encadrés ci-après montrent les fichiers de configuration de BaM, et décrivent les principaux changements par rapport à ceux de BaRatin. Le modèle utilisé comme exemple est le modèle d’orthorectification, qui comprend 3 entrées (coordonnées monde x-y-z) et 2 sorties (coordonnées image colonne-ligne).

## Contrôleur général

Le fichier Config\_BaM.txt permet de définir le workspace et le nom des fichiers de configuration. Le nom du fichier « Config\_BaM.txt » est obligatoire, et ce fichier doit être localisé dans le même dossier que l’exécutable. Tous les autres fichiers de configuration doivent être situés dans le workspace.

Le rôle des différents fichiers de configuration sera détaillé au fil de l’eau.

|  |
| --- |
| Config\_BaM.txt  "C:\BEN\FORGE\BaRatin\trunk\test\BaRatin\_IVF\Tests\BaM\_BiM\" ! workspace  "Config\_RunOptions.txt" ! Config file: run options  "Config\_Model.txt" ! Config file: model  "Config\_Distortion.txt" ! Config file: xtra model information  "Config\_Data.txt" ! Config file: Data  "Config\_RemnantSigma\_Y1.txt","Config\_RemnantSigma\_Y2.txt" ! Config file: Remnant sigma  "Config\_MCMC.txt" ! Config file: MCMC  "Config\_Cooking.txt" ! Config file: cooking of MCMC samples  "Config\_Summary.txt" ! Config file: summary of MCMC samples  "Config\_Residuals.txt" ! Config file: residual diagnostics  "Config\_Pred\_Master.txt" ! Config file: prediction experiments |
| Principales différences par rapport à BaRatin  Dans BaRatin, le workspace devait être tapé à la main dans la console, et le nom des fichiers de configuration était hard-codé. La solution ci-dessus devrait être plus flexible : d’une part il est plus facile d’écrire dans un fichier texte que d’accéder à une console, d’autre part cela donne plus de flexibilité à l’utilisateur pour la gestion de ses fichiers de configuration.  ATTENTION : noter qu’en ligne 6, on doit renseigner autant de fichiers qu’il y a de sorties dans le modèle (ici, 2 : colonnes et lignes). |

## Contrôle des actions que BaM doit exécuter

Ce fichier permet de définir ce que fera l’exécutable. Les actions disponibles sont les suivantes :

* MCMC : simulations MCMC des paramètres à estimer **[implémenté]**
* MCMC cooking : post-processing des simulations MCMC (burn, slim et complétion : voir section 3.5.3) **[partiellement implémenté]**
* MCMC summary : résumé statistique des simulations MCMC  **[implémenté]**
* Residual diagnostics: simulations des sorties du modèle à partir des paramètres maxpost et comparaison avec les données de calibration (voir section 3.5.5) **[implémenté]**
* Predictions : Prédictions et propagation des incertitudes **[implémenté]**

|  |
| --- |
| Config\_RunOptions.txt  .true. ! Do MCMC?  .true. ! Do MCMC cooking?  .true. ! Do MCMC summary?  .true. ! Do Residual diagnostics?  .true. ! Do Predictions? |
| Principales différences par rapport à BaRatin  Fichier similaire mais le détail des actions est légèrement différent |

## Configuration du modèle à caler

Les deux fichiers ci-dessous permettent de définir les propriétés du modèle à caler.

|  |
| --- |
| Config\_Model.txt  "Orthorectification" ! Model ID  3 ! nX: number of input variables  2 ! nY: number of output variables  11 ! nPar: number of parameters theta  "x0" ! Parameter name  10. ! Initial guess  'Gaussian' ! Prior distribution  0.,10000. ! Prior parameters  "y0" ! Parameter name  0. ! Initial guess  'Gaussian' ! Prior distribution  0.,10000. ! Prior parameters  "z0" ! Parameter name  12.6 ! Initial guess  'Gaussian' ! Prior distribution  0.,10000. ! Prior parameters  "azim" ! Parameter name  3.14 ! Initial guess  'Uniform' ! Prior distribution  0., 6.28 ! Prior parameters  "roll" ! Parameter name  0. ! Initial guess  'Gaussian' ! Prior distribution  0., 0.001 ! Prior parameters  "tilt" ! Parameter name  -1.0 ! Initial guess  'Uniform' ! Prior distribution  -3.14159, 3.14159 ! Prior parameters  "f" ! Parameter name  0.05 ! Initial guess  'Gaussian' ! Prior distribution  0.05,0.05 ! Prior parameters  "c0" ! Parameter name  360. ! Initial guess  'Gaussian' ! Prior distribution  360.,10. ! Prior parameters  "l0" ! Parameter name  288. ! Initial guess  'Gaussian' ! Prior distribution  288.,10. ! Prior parameters  "lambda" ! Parameter name  0.00001 ! Initial guess  'Gaussian' ! Prior distribution  0.00001,0.0001 ! Prior parameters  "k" ! Parameter name  1. ! Initial guess  'Gaussian' ! Prior distribution  1.,0.001 ! Prior parameters |
| Principales différences par rapport à BaRatin  En début de fichier, on doit préciser le nombre de variables d’entrée et de sortie.  La spécification des paramètres est ensuite similaire à BaRatin, à une nuance près : dans chaque bloc paramètre, la première ligne donne le nom du paramètre, qui sera repris dans les en-têtes des fichiers de résultat. |

|  |
| --- |
| Config\_Distortion.txt  "None" ! ID  0 ! npar |
| Principales différences par rapport à BaRatin  Nouveau fichier, qui est utilisé pour faire passer toute information complémentaire dont le modèle aurait besoin pour tourner. L’interprétation du contenu de ce fichier est donc complètement différente d’un modèle à l’autre. Ici par exemple (modèle d’orthorectification), on se sert de ce fichier pour faire passer la fonction de distorsion utilisée et le nombre de paramètres associés. Pour une courbe de tarage, on se servirait de ce fichier pour faire passer la matrice de Bonnifait. Pour un modèle de transport sédimentaire, on se servirait de ce fichier pour faire passer les formules choisies pour e.g. le seuil de mise en mouvement, la formule de transport solide, etc. Voir la section 4 pour une description du contenu de ce fichier pour les modèles actuellement disponibles dans BaM. |

## Propriétés de l’inférence

Les deux fichiers ci-dessous permettent de définir les données de calage ainsi que les modèles d’erreur structurelle.

|  |
| --- |
| Config\_Data.txt  "C:\BEN\FORGE\BaRatin\trunk\test\BaRatin\_IVF\Tests\BaM\_BiM\XY.BAD" !!! Absolute path to data file  2 !!! number of header lines  19 !!! Nobs, number of rows in data file (excluding header lines)  5 !!! number of columns in the data file  1,2,3 !!! columns for X (observed inputs) in data file - comma-separated if several  0,0,0 !!! columns for Xsigma (uncertainty in X, EXPRESSED AS A STANDARD DEVIATION - 0 = no-error assumption)  5,4 !!! columns for Y (observed outputs) in data file - comma-separated if several  0,0 !!! columns for Ysigma (uncertainty in Y, EXPRESSED AS A STANDARD DEVIATION - 0 = no-error assumption) |
| Principales différences par rapport à BaRatin  Fichier similaire, mais pour les lignes 5-8, il faut veiller à spécifier autant de colonnes qu’il y a de variables d’entrée pour X et Xsigma (et de même pour Y/Ysigma) |

|  |
| --- |
| Config\_RemnantSigma\_Y1.txt  'Constant' ! Function f used in sdev=f(ysim)  1 ! Number of parameters gamma for f  "remnantStd" ! parameter name  100. ! Initial Guess  'Uniform' ! Prior distribution  0,100000 ! Prior parameters |
| Principales différences par rapport à BaRatin  Fichier quasi-identique, à l’exception de la première ligne du bloc paramètre qui donne à présent le nom du paramètre.  Pour le modèle illustré ici (orthorectification), il existe un second fichier Config\_RemnantSigma\_Y2.txt pour la seconde sortie du modèle, dont le contenu est identique. |

## Configuration des actions de l’exécutable

### MCMC

Simulations MCMC issues de la distribution a posteriori. NOTE : toutes les actions suivantes qui utilisent ces simulations le font après brulage (*burn*) et affinage (*slim*).

|  |
| --- |
| Config\_MCMC.txt  "Results\_MCMC.txt" ! File name  100 ! NAdapt  100 ! Ncycles  0.5 ! BurnFactor  10 ! Nslim  0.1 ! MinMoveRate  0.5 ! MaxMoveRate  0.9 ! DownMult  1.1 ! UpMult  0 ! Mode for setting the initial Std of the jump distribution  \*\*\*\* DEFINITION OF INITIAL JUMP STD \*\*\*\* ! Cosmetics  0.1 ! MultFactor in default mode (ignored in manual mode)  0.1,0.1,0.1 ! RC MultFactor in manual mode (ignored in auto mode)  0.1,0.1 ! Remnant MultFactor in manual mode (ignored in auto mode) |
| Principales différences par rapport à BaRatin  Fichier quasi-identique, mis à part quelques inversions de lignes. |

### MCMC cooking

Le “cooking” des simulations MCMC vise à modifier les simulations brutes (“raw”) via les trois actions suivantes :

1. *Burn* : on efface le début des simulations, en utilisant le facteur BurnFactor spécifié dans Config\_MCMC.txt
2. *Slim* : sur ce qui reste, on ne conserve qu’une ligne toute les X (où X est donné par nslim dans Config\_MCMC.txt)
3. *Complete* : sur ce qui reste, on ajoute un certain nombre de colonnes qui représentent des variables dérivées des paramètres. Quelques exemples :
   1. Pour une courbe de tarage, on ajoutera les paramètres *b* des contrôles supérieurs, calculés par continuité
   2. Pour l’orthorectification, on ajoutera les éléments de la matrice de rotation, calculés à partir des paramètres physiques (angles, focale, etc.)

Pour le moment, les actions *Burn* et *Slim* sont implémentées, l’action *Complete* ne l’est pas.

|  |
| --- |
| Config\_Cooking.txt  "Results\_MCMC\_Cooked.txt" ! Result file |
| Principales différences par rapport à BaRatin  Nouveau fichier. |

### MCMC summary

Résumé des simulations MCMC (après “cooking” éventuel). Les statistiques calculées sont les mêmes que dans BaRatin (moyenne, écart-type, déciles et quartiles, maxpost, etc.).

|  |
| --- |
| Config\_Summary.txt  "Results\_Summary.txt" ! Result file |
| Principales différences par rapport à BaRatin  Nouveau fichier. |

### Residual diagnostics

Cette action vise à réaliser la simulation des variables de sortie du modèle, à partir des entrées observées en calibration. Quelques précisions :

1. Lorsqu’une valeur d’entrée est considérée comme incertaine, on ne force pas le modèle avec la valeur observée () mais plutôt avec la valeur estimée vraie (, où  est l’estimateur maxpost de l’erreur).
2. En comparant les sorties observées et simulées, on peut calculer les résidus 
3. On peut en déduire les résidus standardisés . Si les modèles d’erreurs sont corrects (à la fois pour les erreurs de mesure et l’erreur structurelle), ces résidus standardisés devraient suivre une loi .

|  |
| --- |
| Config\_Residuals.txt  "Results\_Residuals.txt" ! Result file |
| Principales différences par rapport à BaRatin  Nouveau fichier. |

### Prediction

Cette action très générale permet de réaliser des « expériences de prédiction », qui recouvrent plusieurs sous-actions dont :

1. Simulations Maxpost
2. Propagation des incertitudes
3. Simulations a priori

Le fichier de configuration « Master » (voir Config\_Pred\_Master.txt ») recense les expériences de prédiction qui seront réalisées, et donne le fichier de configuration associé à chacune d’entre elles.

Pour chaque expérience de prédiction, un fichier de configuration doit être spécifié. Trois exemples sont fournis ci-dessous, correspondant à une prédiction maxpost, une propagation des incertitudes totales, et une simulation a priori. Chacun de ces fichiers de configuration doit comprendre les informations suivantes :

1. **Chemins d’accès aux spaghettis pour chaque variable d’entrée**: dans BaM, c’est l’utilisateur qui doit fournir les valeurs des variables d’entrée. Ces valeurs peuvent correspondre à une grille (par exemple pour générer une courbe de tarage) ou à une série par exemple temporelle (par exemple pour générer un hydrogramme). De plus, il est possible de fournir des spaghettis pour ces variables d’entrée, i.e. plusieurs réplications représentant l’incertitude (chaque spaghetti étant une colonne dans le fichier de données). C’est une différence majeure par rapport à BaRatin, qui s’occupait lui-même de générer les spaghettis pour les variables d’entrée (hauteurs incertaines). La solution adoptée dans BaM demande un travail supplémentaire pour l’utilisateur, mais est beaucoup plus générale. Par exemple, dans le cas d’un modèle pluie-débit, l’incertitude sur les pluies serait fournie par un simulateur de pluie, qui est bien trop complexe pour être intégré directement dans BaM.
2. **Nombre d’observations dans chaque spaghettis**: nombre de lignes dans chaque fichier de spaghettis (par exemple taille de grille, nombre de pas de temps du limnigramme, etc.). Ce nombre est commun à toutes les variables d’entrée.
3. **Nombre de réplications pour chaque variable d’entrée :** nombre de colonnes dans chaque fichier de spaghettis.
4. **Propagation de l’incertitude paramétrique ?**
5. **Propagation de l’incertitude rémanente ?** A spécifier pour chaque variable de sortie.
6. **Nombre de simulations dans le cas d’une prédiction a priori.** Pour une simulation à partir de la distribution a posteriori, mettre un nombre négatif. Les paramètres seront alors issus des simulations MCMC. Un nombre positif sera interprété comme une demande pour réaliser une simulation à partir de la distribution a priori, et donnera le nombre de simulations réalisées.
7. **Fichiers résultats pour chaque variable de sortie (spaghettis).** Autant de fichiers que de variables de sortie.
8. **Transposer les spaghettis ?** Par défaut les spaghettis sont écrits « à l’envers » dans le fichier résultat : chaque ligne est un spaghetti (pour des raisons liées à l’algorithme de prédiction). Cette action permet de remettre les spaghettis à la verticale.
9. **Calculer les enveloppes ?** Noter que pour l’instant, une option très conservatrice a été implémentée : il suffit d’une unique valeur « impossible » dans un ensemble de spaghettis pour que le calcul des enveloppes renvoie une valeur impossible. Libre à l’utilisateur de retourner au fichier spaghetti brut pour calculer les enveloppes d’une autre manière.
10. **Fichiers résultats pour chaque variable de sortie (enveloppes)**. Autant de fichiers que de variables de sortie.
11. **Afficher une « barre de progrès » en cours de calcul ?**

|  |
| --- |
| Config\_Pred\_Maxpost.txt  'PATH\X1g.txt','PATH\X2g.txt','PATH\X3g.txt' !!! Files containing spaghettis for each input variable (size nX)  1000 !!! Nobs, number of observations per spaghetti (common to all files!)  1,1,1 !!! Nspag, number of spaghettis for each input variable (size nX)  .false. !!! Propagate parametric uncertainty?  .false.,.false. !!! Propagate remnant uncertainty for each output variable? (size nY)  -1 !!! Nsim[prior]  'Y1\_maxpost.spag','Y2\_maxpost.spag' !!! Files containing spaghettis for each output variable (size nY)  .true.,.true. !!! Post-processing: transpose spag file? (size nY)  .false.,.false. !!! Post-processing: create envelops? (size nY)  '' !!! Post-processing: name of envelop files (size nY)  .true. !!! Print progress in console during computations? |
| Principales différences par rapport à BaRatin  Nouveau fichier. |

|  |
| --- |
| Config\_Pred\_Total.txt  'PATH\X1.txt','PATH\X2.txt','PATH\X3.txt' !!! Files containing spaghettis for each input variable (size nX)  350 !!! Nobs, number of observations per spaghetti (common to all files!)  500,500,500 !!! Nspag, number of spaghettis for each input variable (size nX)  .true. !!! Propagate parametric uncertainty?  .true.,true. !!! Propagate remnant uncertainty for each output variable? (size nY)  -1 !!! Nsim[prior]  'Y1\_total.spag','Y2\_total.spag' !!! Files containing spaghettis for each output variable (size nY)  .true.,.true. !!! Post-processing: transpose spag file? (size nY)  .true.,.true. !!! Post-processing: create envelops? (size nY)  'Y1\_total.env','Y2\_total.env' !!! Post-processing: name of envelop files (size nY)  .true. !!! Print progress in console during computations? |
| Principales différences par rapport à BaRatin  Nouveau fichier. |

|  |
| --- |
| Config\_Pred\_Prior.txt  'PATH\X1g.txt','PATH\X2g.txt','PATH\X3g.txt' !!! Files containing spaghettis for each input variable (size nX)  1000 !!! Nobs, number of observations per spaghetti (common to all files!)  1,1,1 !!! Nspag, number of spaghettis for each input variable (size nX)  .true. !!! Propagate parametric uncertainty?  .false. !!! Propagate remnant uncertainty for each output variable? (size nY)  1000 !!! Nsim[prior]  'Y1\_prior.spag','Y2\_prior.spag' !!! Files containing spaghettis for each output variable (size nY)  .true.,.true. !!! Post-processing: transpose spag file? (size nY)  .true.,.true. !!! Post-processing: create envelops? (size nY)  'Y1\_prior.env','Y2\_prior.env' !!! Post-processing: name of envelop files (size nY)  .true. !!! Print progress in console during computations? |
| Principales différences par rapport à BaRatin  Nouveau fichier. |

|  |
| --- |
| Config\_PredMaster.txt  3 !!! Number of prediction experiments  'Config\_Pred\_Maxpost.txt' !!! Config file for experiments - an many lines as the number above  'Config\_Pred\_Total.txt' !!! Config file for experiments - an many lines as the number above  'Config\_Pred\_Prior.txt' !!! Config file for experiments - an many lines as the number above |
| Principales différences par rapport à BaRatin  Nouveau fichier. |

# Annexe : rapide description des modèles disponibles

## Modèle linéaire

Il s’agit d’un simple modèle linéaire de la forme :

|  |  |
| --- | --- |
|  | (4) |

(Note : dans l’équation ci-dessus,  est « repackagé » comme une matrice ).

ID : "Linear"

 : au choix

 : au choix



Extra-information : aucune.

## BaRatin

Il s’agit de BaRatin « classique » dans le formalisme Bonnifaitien (les courbes exotiques ne sont pas implémentées).

ID : "BaRatin"

 (hauteur)

 (débit)

, où  est le nombre de contrôles hydrauliques spécifiés via la matrice de Bonnifait.

Extra-information : la matrice de Bonnifait, exemple de spécification :

|  |
| --- |
| Config\_ControlMatrix.txt  1 0  0 1 |

## GR4J

GR4J historique (cf. Perrin 2000), mais attention le remplissage initial (exprimé en relatif) des réservoirs de production et de routage sont considérés comme 2 paramètres supplémentaires.

ID : "GR4J"

 (précipitation, évapotranspiration potentielle)

 (débit)

 : les 4 paramètres classiques de GR4J + 2 paramètres pour le remplissage initial des réservoirs.

Extra-information : un gros fichier (extrait ci-dessous, voir le fichier complet dans le répertoire BaM\_GR4J) qui ne doit pas être modifié.

|  |
| --- |
| GR4J\_SetupFile.txt  PCO\_MODELSETUP\_FILE\_V1.7  NEW\_BASIC  "GR4J[BRD\_F]" ! modelName  "" ! mdlCmd  2 ! ninput  42 ! nstate  8 ! npar  END\_BASIC  ----------  NEW\_DETALLES  "time" ! indxName  ! \*\*\* iIn, inputName, inScal  1, "rain", 10.0  2, "ETpot", 10.0  ! \*\*\* iState, stateName, stateScal, stateDef, stateLo, stateHi  1, "Vflow ", 10.0, 0.0, 0.0, 5000.0  2, "VrainCor", 10.0, 0.0, 0.0, 5000.0  3, "Prod, -", 10.0, 100.0, 0.0, 500.0  4, "Rout, -", 10.0, 5.0, 0.0, 500.0  5, "F, -", 10.0, 5.0, 0.0, 500.0  6, "QR ", 10.0, 0.0, 0.0, 5000.0  7, "QD ", 10.0, 0.0, 0.0, 5000.0  8, "Es ", 10.0, 5.0, 0.0, 500.0  9, "Ei ", 10.0, 5.0, 0.0, 500.0  10, "Ps ", 10.0, 5.0, 0.0, 500.0  11, "Pr ", 10.0, 5.0, 0.0, 500.0  12, "Perc ", 10.0, 5.0, 0.0, 500.0  13, "uh1\_01", 10.0, 0.0, 0.0, 1.0  14, "uh1\_02", 10.0, 0.0, 0.0, 1.0  15, "uh1\_03", 10.0, 0.0, 0.0, 1.0  16, "uh1\_04", 10.0, 0.0, 0.0, 1.0  17, "uh1\_05", 10.0, 0.0, 0.0, 1.0  18, "uh1\_06", 10.0, 0.0, 0.0, 1.0  19, "uh1\_07", 10.0, 0.0, 0.0, 1.0  20, "uh1\_08", 10.0, 0.0, 0.0, 1.0  Etc. |

## Orthorectification

Utilise le formalisme et les conventions de la thèse d’Alexandre Hauet, à une exception près : le dernier paramètre n’est pas la taille horizontale des pixels, mais plutôt l’ « aspect ratio » par rapport à la taille verticale.

ID : "Orthorectification"

 (coordonnées monde, x-y-z)

 (coordonnées image, colonne-ligne)

 : 6 paramètres extrinsèques + 5 paramètres intrinsèques.

Extra-information : fonction de distorsion, exemple de spécification :

|  |
| --- |
| Config\_Distortion.txt  "None" ! ID  0 ! npar |

Les fonctions de distorsion disponibles sont :

* "None" : pas de distorsion, 0 paramètres
* "Radial" : distorsion radiale . Le degré du polynôme se déduit du nombre de paramètres (2 paramètres => degré 4, etc.)

## Transport sédimentaire

Bouh que c’est compliqué… Formules de transport sédimentaire classiques, avec tout un tas d’options pour gérer les différentes configurations possibles.

ID : "Sediment"

 : variable suivant les options choisies, mais à minima 2 (hauteur d’eau  et pente d’énergie )

 (débit solide volumique par unité de largeur  (m²/s), DIMENSIONNEL)

 : variable suivant les options choisies

Extra-information : options choisies pour construire la formule de transport, exemple de spécification :

|  |
| --- |
| Config\_Options.txt  "Camenen" ! base formula  "Soulsby" ! Critical Shear Stress formula  "FIX" ! CSS Coefficient option  "IN","IN","FIX","FIX" ! pseudo-parameters options  0.000506,2.65,0.000001,9.81 ! pseudo-par values |

Explications

Une formule de transport est obtenue par la combinaison des éléments suivants :

1. Une formule de base, choisie dans le catalogue suivant (avec dans toutes ces formules la constante d’adimensionnalisation et  la contrainte adimensionnelle):
   1. "MPM" (Meyer-Peter & Mueller) : 
   2. "Camenen": 
   3. "Nielsen": 
   4. "Smart": 
2. Une formule pour la contrainte de mise en mouvement, choisie dans le catalogue suivant (avec dans ces formules  le diamètre sédimentologique) :
   1. "Constant": Dans ce cas,  est traité comme un paramètre additionnel qui rejoint la liste des 
   2. "Soulsby": 
   3. "SoulsbyWhitehouse": 

Dans le fichier Config\_Options.txt, les deux premières lignes spécifient les formules de base et de contrainte de mise en mouvement, respectivement.

La troisième ligne gère le traitement des coefficients (0.3, 1.2, 0.055, 0.02 et 0.24) dans les formules 2b et 2c. "FIX" signifie que les coefficients sont fixés aux valeurs numériques ci-dessus. "PAR" signifie que ces coefficients vont être considérés comme des paramètres inconnus à estimer.

Les lignes 4 et 5 gèrent le traitement des « pseudo-paramètres »  (diamètre des grains, le *d*50 en général),  (densité des grains),  (viscosité cinématique de l’eau) et  (gravité). En effet suivant les jeux de données à analyser, chacun de ces pseudo-paramètres  peut être traitée comme une constante numérique fixée, comme une constante numérique imparfaitement connue, ou comme une variable d’entrée du modèle. Pour chaque pseudo-paramètre, on peut donc choisir entre les trois options suivantes :

1. "FIX": le pseudo-paramètre est une constante numérique parfaitement connue, dont la valeur est donnée en ligne 5.
2. "PAR": le pseudo-paramètre est une constante numérique imparfaitement connue. On traite donc le pseudo-paramètre comme un paramètre inconnu à estimer.
3. "IN": le pseudo-paramètre varie d’une ligne à l’autre dans le fichier de données, on le considère donc comme une variable d’entrée.

Par exemple :

* On considèrera généralement  comme "FIX", avec une valeur fixée à 9.81. Mais si les données viennent de divers endroits du globe et qu’on a envie de prendre en compte la variation de gravité, on pourra en faire une variable d’entrée (et la valeur de  devra être spécifiée dans le tableau de données pour chaque ligne).
* On considèrera généralement  comme "FIX", avec une valeur fixée à 10-6. Mais si on n’est pas trop sûr de cette valeur, on peut considérer  comme "PAR" et spécifier un a priori qui reflète l’incertitude sur cette valeur.
* On considèrera  et  comme "IN" dès que ces variables ne sont pas constantes pour le jeu de données analysées. Si ces variables sont constantes, on pourra les considérer comme "FIX" (valeur constante connue parfaitement) ou "PAR" (valeur constante imparfaitement connue).

Pour finir, noter que les paramètres à estimer doivent être spécifiés dans l’ordre suivant dans le fichier Config\_Model.txt :

1. Paramètres de la formule de base
2.  si la formule de contrainte de mise en mouvement est "Constant"
3. Pseudo-paramètres traités avec l’option "PAR" (en respectant l’ordre ,,,)
4. Coefficients de la formule de contrainte de mise en mouvement traités avec l’option "PAR"

## Stage-Fall-Discharge (SFD)

Il s’agit des modèles SFD étudiés dans la thèse de Valentin Mansanarez.

ID : "SFD"

 (hauteur amont h1 et aval h2)

 (débit)

 : variable suivant les options choisies.

Extra-information : modèle utilisé, et paramètres de l’algorithme de Newton utilisé pour calculer la hauteur de transition (équation implicite)

|  |
| --- |
| Config\_Options.txt  "SFD\_Val\_General" ! ID of the SFD rating curve  20. ! Newton - upper bound  5. ! Newton - xscale  1000. ! Newton - fscale  0.00001 ! Newton - xtol  0.0000000001 ! Newton - ftol  100 ! Newton - maxiter |

Modèles SFD disponibles

**SFD\_Val\_General** : modèle à 8 paramètres

