Note de travail sur l'utilisation de l'algorithme de minimisation DFO sur l'application d'assimilation de données avec MASCARET via PALM

S. Ricci, A. Piacentini, T. Morel, A. Tröltzsch
URA CERFACS/CNRS No1875 - Toulouse - France

Objectif du document

On décrit ici la partie technique de mise en place d'une maquette d'assimilation de données sur le code d'hydraulique 1D MASCARET utilisant un algorithme 3D-VAR où la minimisation est faite par l'algorithme DFO ne nécessitant pas le calcul du gradient de la fonction coût.

1 Introduction : Justification de la méthode

La problématique de l'assimilation de données conduit à la formulation de la fonction coût J (Eq. 1).

$$J(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}^b)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}^b) + \frac{1}{2}(\mathbf{y}^o - H(\mathbf{x}))^T\mathbf{R}^{-1}(\mathbf{y}^o - H(\mathbf{x})). \quad (1)$$

La solution optimale du problème d'assimilation $\mathbf{x}^a$ est de variance minimum et minimise J de sorte que $\nabla(J(\mathbf{x}^a)) = 0$.

L'analyse $\mathbf{x}^a$ peut être approximée par la solution du BLUE (Eq. 2) :

$$\mathbf{x}^a_{BLUE} = \mathbf{x}^b + \mathbf{B}\mathbf{H}^T(\mathbf{H}\mathbf{B}\mathbf{H}^T + \mathbf{R})^{-1}(\mathbf{y}^o - H(\mathbf{x}^b)) \quad (2)$$

$\mathbf{x}^a_{BLUE}$ est véritablement le minimum de J si J est quadratique, c'est à dire, si l'opérateur d'observation H est linéaire. Dans le cas contraire, $\mathbf{x}^a_{BLUE}$ est le minimum de J_{lin} (Eq. 3) :

$$J_{lin}(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}^b)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}^b)+\frac{1}{2}(\mathbf{y}^o-H(\mathbf{x}^b)-\mathbf{H}(\mathbf{x}-\mathbf{x}^b))^T\mathbf{R}^{-1}(\mathbf{y}^o-H(\mathbf{x}^b)-\mathbf{H}(\mathbf{x}-\mathbf{x}^b)). \quad (3)$$

La différence entre ces deux solutions est présentée en Fig. 1 pour le cas du contrôle d'un paramètre K représentant le coefficient de diffusion d'un processus mono-dimensionnel.

La solution optimale $\mathbf{x}^a$ doit être obtenue en minimisant J. Classiquement, un minimiseur requiert le calcul de $J(\mathbf{x})$ et $\nabla(J(\mathbf{x}))$ à chaque itération. Le

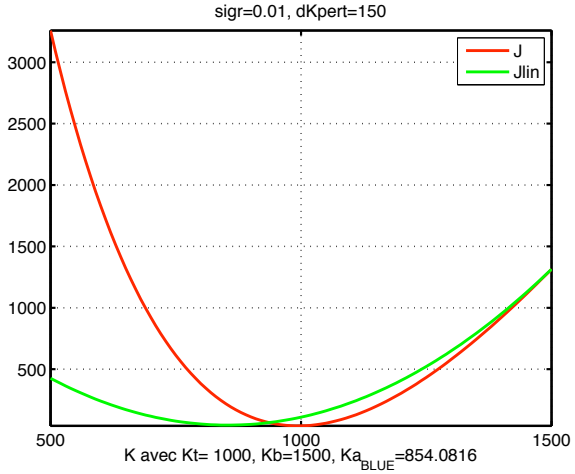


Figure 1: Fonctions coût J (courbe rouge) et J_{lin} (courbe verte) pour le contrôle du coefficient de diffusion avec une physique mono-dimensionnelle

calcul de $\nabla(J(\mathbf{x}))$ (Eq. 4) requiert la formulation de la Jacobienne de H notée $\mathbf{H}$. Cette Jacobienne est calculée autour du point courant $\mathbf{x}$. L'expression de $\nabla(J(\mathbf{x}))$ est aisée si on dispose de la formulaton analytique de $\nabla(J)$ ou d'un code de calcul (le code tangent du code direct) permettant l'évaluation de $\nabla(J)$ en tout $\mathbf{x}$. Dans le cas contraire, on choisit généralement d'évaluer $\mathbf{H}$ par différences finies, ce qui nécessite de nombreuses intégrations du code direct (au moins une par variable de contrôle, et ce à chaque itération du minimiseur).

$$\nabla(J(\mathbf{x})) = \mathbf{B}^{-1}(\mathbf{x} - \mathbf{x}^b) + \mathbf{H}^T\mathbf{R}^{-1}(\mathbf{y}^o - H(\mathbf{x})) \tag{4}$$

La maquette d'assimilation de données sur le code MASCARET permet la correction instantannée de l'état hydraulique (hauteur d'eau et débit) ainsi que la correction des lois hydrauliques décrivant les conditions aux limites (forçage amont ou latéral). Pour la première correction le passage entre l'espace de contrôle et l'espace des observations se fait via une simple sélection de point. L'opérateur d'observation H est donc linéaire, le choix d'un algorithme BLUE pour cette approche est adapté et conduit à la même solution qu'un algorithme variationnel. La correction des lois hydrauliques se fait via trois paramètres a, b, c tels que $\tilde{q(t)} = a \times q(t-c)+b$. L'opération qui permet de passer de l'espace des paramètres à l'espace des observations consiste ici en une intégration de MASCARET suivie d'une sélection de point. Cette opération est non linéaire par rapport à a, b, c. Néanmoins, il a été vérifié que dans un intervalle restreint, il était possible de faire une hypothèse de linéarité locale autour d'un état de référence (Eq.5) :

$$H(\mathbf{x}) \approx H(\mathbf{x}^b) + \mathbf{H}(\mathbf{x} - \mathbf{x}^b) \tag{5}$$

où $\mathbf{H}$ est calculé autour de $\mathbf{x}^b$, par différences finies. Cette formulation permet donc d'appliquer l'algorithme du BLUE. Dans cette étude, on ne dispose pas du code tangent de MASCARET par rapport aux paramètres de contrôle. Pour exemple, si on cale n paramètres (3 par loi hydraulique), chaque estimation de $\mathbf{H}$ requiert $2n$ intégrations du code soit un coût de $2n \times Nb_{iter}$. Sous ces hypothèses, l'analyse résultant du BLUE conduit à une amélioration notable des simulations permettant d'améliorer les scores de prévision des crues comme présenté dans les documents [3], [2].

On propose ici la mise en œuvre d'un algorithme d'assimilation variationnel 3D-Var permettant de réduire J et non J_{lin} en utilisant un minimiseur ne nécessitant pas l'évaluation du gradient de J. On attend une validation ou invalidation de l'hypothèse de linéarité pour la correction des lois hydrauliques. Les développements permettent aussi la mise en place d'un algorithme utile pour une extension de l'espace de contrôle à d'autres paramètres avec des physiques non linéaires (par exemple les coefficients de Stricklers).

2 Présentation succinte de l'algorithme DFO

L'algorithme de minimisation utilisé dans cette étude a été implémenté au CERFACS, dans l'équipe ALGO, au cours de la thèse de A. Tröltzsch. Ces travaux sont référencés en [1]. L'algorithme de Derivative Free Optimization (DFO) s'appuie sur plusieurs évaluations de la fonction f à minimiser pour trouver un modèle polynomial approximant la fonction. C'est le gradient de ce modèle qui est évalué et non celui de la fonction f pour fournir un état intermédaire. Ce processus est itératif. Le modèle est minimisé à l'intérieur d'une région de confiance. Pour que le modèle soit bien défini, le set d'interpolation doit couvrir suffisament d'espace autour de l'itéré courant. Contrairement aux méthodes d'optimisation courantes, cette méthode ne permet pas de couvrir l'espace complet des solutions. Pour que la solution soit satisfaisante, au fil des itérations, l'algorithme doit maintenir une bonne géométrie pour l'ensemble du set d'interpolation.

Soit le problème d'optimisation :

$$min_{x\in\mathbb{R}^n} f(x) \quad with \quad l \leq x \leq u, \tag{6}$$

où f est une fonction non linéaire de $\mathbb{R}^n$ dans $\mathbb{R}$ et l et u sont des vecteurs (possiblement infinis) bornant x. A chaque itération, un modèle de la forme

$$m_k(x_k + s) = f(x_k) + g_k^T \ s + \frac{1}{2} s^T H_k \ s \tag{7}$$

(où g_k et H_k sont les gradient et la Hessienne du modèle) est minimisé dans une région de confiance

$$B_{\inf}(x_k, \Delta_k) = \{x \in \mathbb{R}^n | ||x - x_k||_{\inf} \leq \Delta_k\} \tag{8}$$

où $||_{\text{inf}}$ dénote la norme infinie. Cette minimisation conduit au point $x_k + s_k$ qui est accepté comme nouvel itéré si la réduction de la fonction objective f est plus importante que la réduction du modèle, c'est à dire que le ratio

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m(x_k) - m(x_k + s_k)} \tag{9}$$

est supérieur à la constante $\eta > 0$. Dans ce cas, le modèle est mis à jour et la région de confiance est agrandie. Si $\rho_k \leq \eta$, le point est rejeté et la région de confiance est réduite.

Le modèle m est déterminé en interpolant les valeurs connues de la fonction objective pour le set de points $Y_k = \{y^1, y^2, \cdots, y^p\} \in \mathbb{R}^n$, c'est à dire que

$$m_k(y) = f(y) forall y \in Y_k. \tag{10}$$

En considérant la base polynomiale $\phi = \{\phi_1(x), \phi_2(x), \cdots, \phi_p(x)\}$ comme un set de p polynomes de degrés $\leq d$ qui couvre l'espace des polynomes P_n^d de degrés $\leq d$ dans $\mathbb{R}^n$, le modèle $m(x)$ s'écrit

$$m_k(x) = \sum_{j=1}^{p} \alpha_j \phi_j(x), \tag{11}$$

où α_j sont des coefficients réels. Etant donné le set d'interpolatont Y_k , les coefficients α_j du vecteur α_ϕ sont déterminés en résolvant le système linéaire $M(\phi, Y_k)\alpha_\phi = f(Y_k)$ où

$$M = \begin{pmatrix} \phi_1(y^1) & \phi_2(y^1) & \cdots & \phi_p(y^1) \\ \phi_1(y^2) & \phi_2(y^2) & \cdots & \phi_p(y^2) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_1(y^p) & \phi_2(y^p) & \cdots & \phi_p(y^p) \end{pmatrix}$$

Le choix du set d'interpolation Y_k est crucial car M devient mal conditionné ou singulier si les points de Y_k sont mal positionnés.

L'utilisation de l'algorithme DFO requiert la spécifiaction de la fonction objective $f(x)$, d'un point de départ pour la minimisation y_0 à partir duquel le set d'interpolation Y_0 est calculé, ainsi que le rayon de la région de confiance et une précision pour le critère d'arrêt de la minimisation.

3 Utilisation d'unité Matlab (Octave) sous PALM

L'interfaçage entre une unité Matlab (Octave) a été rendue possible par l'équipe PALM du CERFACS (http://www.cerfacs.fr/globc/PALM_WEB/). Cette

fonctionnalité s'appuie sur les développements permettant d'écrire des unités PALM dans des langages interprétés comme Python, tcl/tk ou Perl. Elle a simplement été adaptée pour l'écriture d'unités en Octave qui est un interpréteur domaine public du logiciel Matlab.

L'interfaçage des routines PALM pour Octave passe par le chargement d'une librairie dynamique liée à la librairie PALM dans l'interpréteur Octave. La construction de cette librairie dynamique s'appuie sur l'outil SWIG. La mise en œuvre de SWIG se fait par un fichier d'interface (palm.i) qui donne à l'outil les prototypes des routines PALM à interfacer. Dans ce fichier d'interface, on déclare également des fonctions permettant de manipuler des types simples comme des pointeurs sur des entiers, des doubles précision, etc. On déclare également des fonctions permettant de manipuler des tableaux contenant des variables de ces types simples. Avec ces fonctions, il est aisé pour l'utilisateur de passer d'une variable Octave à une variable C et réciproquement. Seules les variables C créées par ces fonctions sont utilisables directement dans les primitives PALM appelées depuis Octave. A titre d'exemple, on mentionne ici la primitive palm.PALM_Print permettant d'afficher une chaine de caractères dans les fichiers de sortie de PALM (.log).

Les unités Octave sont dotées d'une carte d'indentité décrivant l'unité et les objets en entrée et sortie de l'unité, notons que le caractère # doit être utilisé pour mettre ces lignes sous forme de commentaire (car le % n'est pas pris en compte par PrePALM). Les déclarations de variables C et l'appel des primitives permettant d'échanger les objets sont écrites dans le script Octave.

Le lancement de l'unité s'appuie sur un fichier de commande permettant de lancer l'unité octave (octave main.m &), le nom de ce fichier de commande est donné dans la carte d'identité de l'unité :

```
PALM_UNIT -name run_bcdfo
          - functions SH run_bcdfo.sh&
          - comment run_bcdfo
```

Dans l'unité Octave, la commande :
palm;
Permet de charger la librairie dynamique.
La commande :
err = palm.PALM_Mpi_init();
Permet d'initialiser le contexte MPI (équivalent à MPI_init, rappelons que PALM est basé sur la librairie de message passing MPI pour gérer les échanges entre les composants).
La ligne :
err = palm.PALM_Connect();
Permet d'établir dynamiquement la connexion MPI entre le driver de PALM

et l'interpréteur Octave. De manière symétrique la déconnexion de PALM et la sortie de MPI se font via les commandes
err = palm.PALM_Disconnect()
err = palm.PALM_Mpi_finalize()

La déclaration d'une variable utilise les fonctions décrites dans le module d'interface palm.i, la ligne :
time_p = palm.new_int(palm.PL_NO_TIME);
déclare un pointeur sur entier,
ila_shape = palm.new_array_int(1);
déclare un tableau contenant un entier,
dla_x = palm.new_array_double(10);
déclare un tableau de 10 doubles précision,
nx = palm.get_int(ila_shape,0);
permet de récupérer la valeur du premier élément contenu dans le tableau d'entiers ila_shape dans la variable Octave nx,
err = palm.PALM_Get('NULL','x',time_p,tag_p,dla_x);
permet d'effectuer le PALM_Get de l'objet x d'espace NULL.

Si les appels aux primitives PALM doivent se faire dans des sous-programmes d'Octave (function Octave), l'objet palm doit être passé en argument de la fonction octave :
[result_main] = my_routine(palm,x)
function [result_my_routine] = my_routine(palm, x)

L'inclusion de routines Octave dans une application PALM n'a pas d'incidence sur le Makefile dans la mesure où les unités Octave ne nécessitent pas de compilation. En revanche, après l'exécution du Makefile, il est nécessaire de lancer l'outil SWIG pour construire la librairie dynamique adaptée à Octave. Le fichier make_swig permet d'effectuer ce lien avec les librairies palm et Octave. Pour lancer la compilation on entre :
make -f make_swig

Enfin, dans le schéma Prepalm, le mode d'exécution des unités Octave doit être mis à EXTERN-TO-CONNECT.

4 Description de la maquette

Dans notre application, nous avons instrumenté l'algorithme de DFO afin de le faire fonctionner pour le contrôle des lois hydrauliques de MASCARET. Les deux étapes principales de la construction de la maquette sont :

- la formulation de la fonction coût dans le code d'assimilation,
- l'instrumentation du code DFO pour l'évaluation de la fonction coût à chaque itération.

La formulation de $J(\mathbf{x})$ utilise en grande partie les objets mathématiques formulés pour l'algorithme du BLUE. Le programme MASCARET a seulement été adapté pour retourner (avec des Palm_Put)non pas le vecteur d'innovation $\mathbf{d}$ mais les vecteurs $\mathbf{y}^o$ et $H(\mathbf{x})$. Les matrices de covariances ainsi que l'ébauche sont communiquées depuis la routine d'initialmisaton read_damocles.f90. Une routine fortran Cstfct.f90 formule les parties d'ébauche et d'observation de la fonction coût et retourne J. Pour cette maquette, l'instrumentation du code MASCARET n'a donc pas nécessité de modification majeure sous l'option KEY_CALAGE_PALM. Dans PrePalm, ces actions sont séquentiellement parcourue dans la branche Branche_modele en Start ON (on reviendra sur ce point plus tard).

Le minimiseur DFO est composé d'une routine principale run_bcdo.m qui appelle la subroutine bcdo.m avec une liste d'arguments dont le point de départ de la minimisation, le rayon de la région de confiance, les bornes de la minimisation au besoin, la précision, le nombre d'itérations et d'évaluations maximum ainsi qu'un indicateur de la fonction analytique f à minimiser. La routine run_bcdfo.m a été déclarée comme une unité PALM, elle est instrumentée par un Palm_Get de l'ébauche (point de départ) communiqué par read_damocles.f90. Elle communique aussi par un Palm_Put, un flag (valeur à 1) pour signaler que la minimisation est terminée. La routine décrivant la fonction objective est elle instrumentée par 4 communications :

- un Palm_Put du flag (valeur à 0) pour signaler que la minimisation n'est pas terminée,
- un Palm_Put du point d'évaluation courant $\mathbf{x}$ communiqué vers la branche Branche_modele pour l'évaluation de $J(\mathbf{x})$,
- un Palm_Get d'un flag qui permet de quitter le minimiseur si il n'y a pas lieu de faire la minimisation (typiquement si toutes les observations sont rejetées, l'analyse est alors égale à l'ébauche)
- un Palm_Get de $J(\mathbf{x})$ communiqué par la Branche_modele.

Le minimiseur est placé dans la branche Branche_DFO à la suite d'un appel à l'unité read_damocles.f90 pour initialisation.

Entre les 2 branches, un Palm_Get de branche dans Branche_modele sur le flag permet de sortir d'une boucle WHILE qui effectue les évaluations de $J(\mathbf{x})$ tant que la minimisation n'est pas terminée. Ce get a été inclu dans un bloc pour qu'il ne soit pas effectué par le driver et puisse rester en attente.

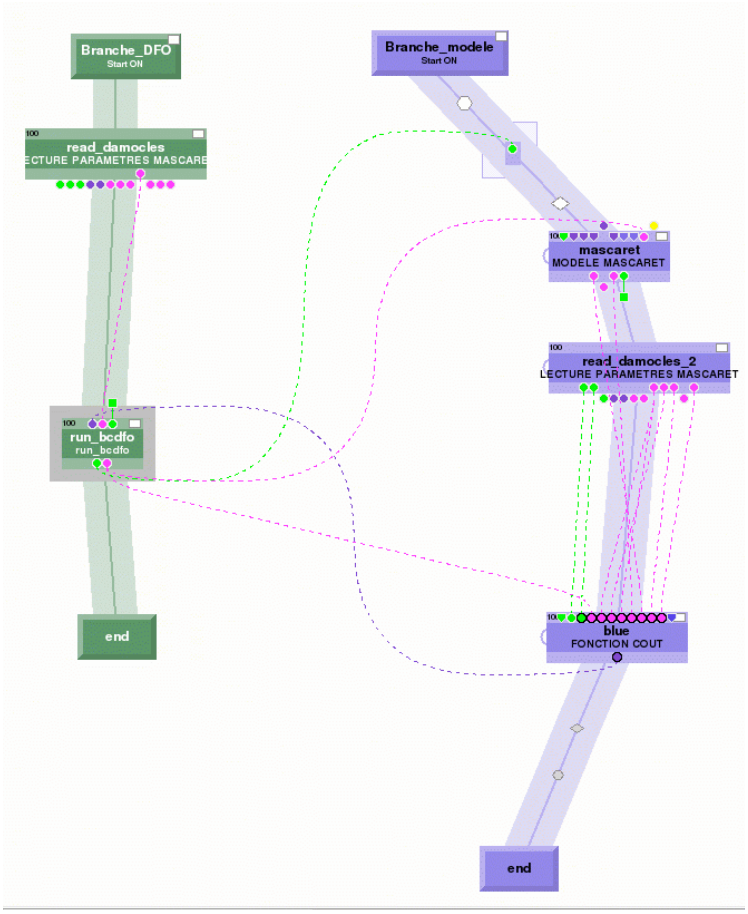


Figure 2: Schéma PrePALM de la maquette 3D-Var avec minimisation par l'algorithme DFO pour le contrôle des lois hydrauliques amonts avec MASCARET

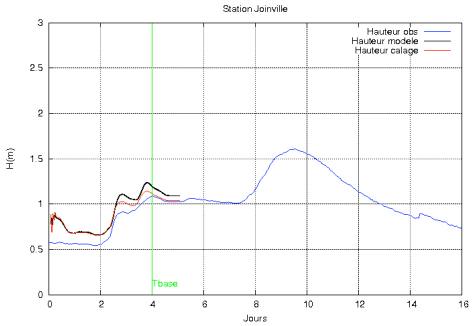
Les 2 branches sont en start ON et partagent les ressources CPU du calculateur séquentiellement : pendant qu'un processus évalue la fonction coût, le minimiseur est en attente et inversement. Ce fonctionnement ne permet pas de libérer les ressources de la branche Branche_modele lorsque le minimiseur travaille, néanmoins, dans notre maquette, à cet instant aucune autre action n'est à accomplir. Le fonctionnement avec les 2 branches actives en permanence est imposé par la nature du minimiseur qui n'est pas *reverse communication* : nous n'avons en effet pas accès à une routine qui effectue une itération à la fois comme cela est le cas avec les minimiseurs utilisés dans le cadre de la formation PALM par exemple. Sans cette contrainte, on aurait pu appeler la branche Branche_modele dans le branche Branche_DFO au besoin, les ressources auraient été libérées après chaque évaluation de $J(\mathbf{x})$. Une optimisation consisterai à passer l'algorithme DFO en reverse communication de l'algorithme, ceci n'est pas envisagé pour l'instant.

5 Cas test et résultats préliminaires

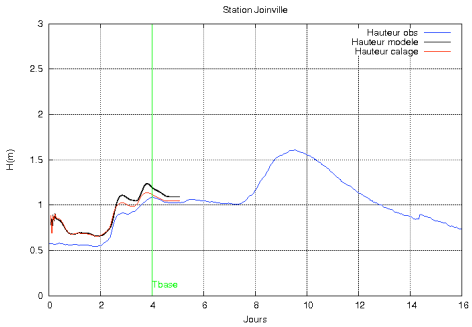
L'algorithme variationnel avec le miniseur DFO a été mis en œuvre sur un cas test dans le but de valider ses résultats. A ce stade, l'ensemble des analyses produites avec le BLUE n'a pas été reproduit avec le 3D-Var/DFO. Nous examinons ici l'analyse au premier temps de base de l'événement 05 sur le bassin versant de Marne Vallage. Habituellement, sur les 2 jours précedent le temps de base, les observations sont seuillées car en dessous du seuil jaune. Néanmoins, pour le cas test, on supprime ce critère de rejet des observations pour le BLUE et pour le 3D-Var/DFO. Au delà du temps de base, on effectue un jour de prévision. L'analyse est effectuée avec les écarts types et les valeurs d'ébauche pour les paramètres a, b, c habituellement spécifiés dans le fichier Parametres_Mascaret.cas. On a notamment $(a^b, b^b, c^b) = (1, 0, 0)$. On note ici, que les paramètres a, b, c sont de dimension différentes. Le rayon de la région de confiance devrait donc être différent pour chaque paramètre. Cette fonctionnalité n'étant pas disponible dans l'algorithme dont nous disposons, nous avons choisi de redimensionner les variables à la sortie de chaque itération du minimiseur (7200 s pour c).

Les valeurs analysées obtenues avec l'algorithme du BLUE sont
$\mathbf{x}^a_{BLUE} = (0.796, 1.17, 5833, 0.98, 1.09, -2313)$.
Celles obtenues avec le 3D-Var/DFO sont
$\mathbf{x}^a_{3D-Var/DFO} = (0.8, 1.2, 7344, 0.99, 1.02, -4320)$.
L'algorithme DFO a été poussé à 400 itérations, néanmoins, la valeur de l'itéré n'évolue plus bien avant ce nombre. De plus amples tests (notamment sur le critère de précision) devront être faits pour permettre une utilisation optimale du minimiseur.

Les résultats de la simulation avec les lois hydrauliques corrigées selon les analyses $\mathbf{x}^a_{BLUE}$ et $\mathbf{x}^a_{3D-Var/DFO}$ sont présentés en Fig. 3 (a) et (b). On note que les corrrections ménent à des simulations identiques. Sur ce cas test, il apparaît que l'hypothèse de linéarité semble valide et que la solution du BLUE est une bonne approximation de la solution optimale.



(a)



(b)

Figure 3: Hauteur d'eau simulée à Joinville pour le modèle libre (courbe noire), le modèle avec assimilation (courbe rouge) (a) algorithme du BLUE (b) algorithme 3D-Var/DFO, et hauteur d'eau observée (courbe bleu).

References

[1] S. Gratton, Ph.L. Toint, and A. Tröltzsch. An active-set trust-region method for derivative-free nonlinear bound-constrained optimization. *Technical Report CERFACS*, TR/PA/10/70, 2010.

[2] A. Piacentini, O. Thual, G. Jonville, C. Ivanoff, S. Ricciand S. Massart, and E. Le Pape. Assimilation de données en hydraulique, maquette pour le modèle mascaret. *Technical Report CERFACS*, TR/CMGC/10/16, 2010.

[3] S. Ricci, A. Piacentini, O. Thual, E. Le Pape, and G. Jonville. Correction of upstream flow and hydraulic state with data assimilation in the context of flood forecasting. *Submitted to HESS*, 2010.