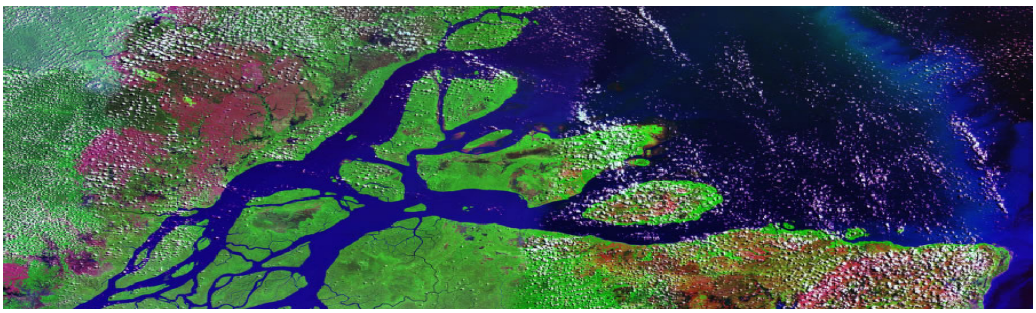
École Nationale de la Météorologie
Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique
Encadrantes : M. Mouffe[1,2] et S. Ricci[1]

[1] CERFACS/URA1875, [2] CNES

Clément Doche, Jean Mathorel et Tom Nicolau
10 Février 2012

Optimisation de paramètres d'un modèle d'hydrologie globale



NASA©

Projet de modélisation de deuxième année de l'École Nationale de Météorologie
Année scolaire 2011-2012

Résumé

Le modèle HyMAP est un modèle d'hydrologie globale qui permet de simuler le débit et les hauteurs d'eau des rivières à grande échelle. Ce modèle repose sur une paramétrisation de propriétés physiques et géométriques des rivières telles que la rugosité et les dimensions du lit ou le temps de transfert de l'eau par drainage et écoulement de surface. Ces propriétés étant mal connues, il est nécessaire de les optimiser en utilisant des d'observations (données altimétriques fournies par ENVISAT). Leur calibration est à ce jour effectuée par le biais d'une optimisation multi-critères avec l'algorithme sans gradient évolutionnaire MOCOM-UA (Multi-Objective Complex Optimization Method Algorithm, Yapo *et al.*, 1997), et nécessite un grand nombre d'évaluations des fonctions objectif, ce qui entraîne un fort coût de calcul. L'objectif de ce projet de modélisation est donc de coupler une méthode d'optimisation moins coûteuse en temps de calcul avec le modèle HyMAP, la méthode BC-DFO (méthode mono-objectif locale avec contraintes de bornes). Ce couplage s'est avéré effectivement plus léger en temps de calcul et la qualité de l'optimisation est conservée comme le montrent les résultats. Cependant, un risque de converger vers un minimum local éloigné du minimum global existe avec l'optimiseur BC-DFO. Le projet s'est aussi orienté vers l'assimilation de données appliquée aux paramètres du modèle HyMAP *via* l'optimiseur BC-DFO.

Mots-clés : optimisation de paramètres, HyMAP, MOCOM-UA, BC-DFO, hydrologie, assimilation de données, ENVISAT, couplage de codes.

Sommaire

Introduction

Présentation du CERFACS

Le **CERFACS** (Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique) est un organisme de recherche travaillant sur les simulations numériques et les solutions algorithmiques dans un grand nombre de domaines scientifiques et techniques. Pour mener à bien ses activités, le CERFACS utilise des moyens de calcul à haute performance (HPC).

Il est dirigé par un Conseil de Gérance qui représente ses actionnaires, et bénéficie des recommandations du Conseil Scientifique. Les sept actionnaires du CERFACS sont : le **CNES** (Centre National d'Etudes Spatiales) ; **EADS** (European Aeronautic and Defense Space Company) ; **EDF** (Electricité de France) ; **Météo-France** ; l'**ONERA** (Office National d'Etudes et de Recherches Aérospatiales) ; **SAFRAN**, équipementier international de hautes technologies, et **TOTAL**.

Le CERFACS emploie environ 115 personnes, dont plus de 95 chercheurs et ingénieurs, au travers d'équipes interdisciplinaires comprenant physiciens, chercheurs en mathématiques appliquées, analystes numériques, et ingénieurs programmeurs.

Les principaux domaines de recherche du CERFACS sont : les algorithmes parallèles, le couplage de codes numériques, l'aérodynamique, les turbines à gaz, la combustion, le climat, l'impact environnemental, l'assimilation de données, les champs électromagnétiques et acoustiques, l'analyse numérique et l'optimisation (Fig. 1).

Le CERFACS est associé au **CNRS** (Centre National de Recherche Scientifique) au travers de l'URA1875 qui englobe les activités des équipes **GLOBC** (GLOBal Change & climate modeling) et **PAE** (Aviation Environment). Une collaboration avec l'**INRIA** (Institut National de Recherche en Informatique et Automatique) est formalisée par le groupe **HIEPACS** qui regroupe des activités de l'équipe **ALGO** (ALGOrithme et calcul).

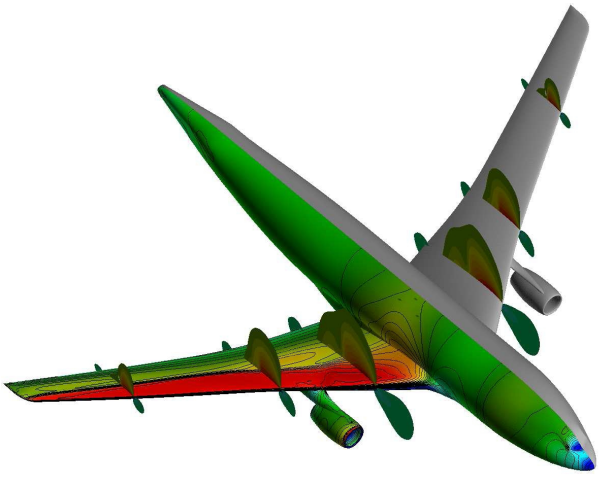


FIGURE 1 – Optimisation de la traînée d'un avion (*cerfacs.fr*)

Le CERFACS participe également aux programmes de **TVE** (Terre Vivante et Espace) et d'**AESE** (Aéronautique, Espace et Systèmes Embarqués), et est membre du pôle **RTRA/STAE** (Réseau Thématique de Recherche Avancée Sciences et Technologies pour l'Aéronautique et l'Espace).

Présentation du projet de modélisation

Le modèle **HyMAP** (Hydrological Modeling and Analysis Platform, Getirana *et al.*, 2011) est un schéma d'écoulement de l'eau de surface à l'échelle globale. Il permet de simuler quotidiennement le niveau, le débit et le stockage de l'eau dans les rivières ainsi que la surface des plaines inondables à une résolution spatiale de 0.25 °sur le globe (environ 25 km).

Pour décrire correctement le comportement et la géométrie du lit d'une rivière, le modèle doit être paramétré au mieux. Pour cela, il est nécessaire de définir une fonction objectif qui sera minimisée par un algorithme (optimiseur). Ici, cette fonction quantifie l'écart entre les simulations du modèle et les observations fournies par le satellite **ENVISAT**. Ce satellite mesure des hauteurs d'eau le long d'une fine trace. L'écart entre les simulations et les observations peut être exprimé de différentes façons (par exemple le coefficient Nash-Sutcliffe).

Dans le cas de HyMAP, divers paramètres sont utilisés afin de simuler le débit et la hauteur d'eau (largeur et hauteur du lit, rugosité induisant des forces de frottements, *etc*). Il faut donc optimiser ces paramètres pour améliorer les simulations. Actuellement, les paramètres d'HyMAP sont optimisés *via* la méthode **MOCOM-UA** (Multi-Objective Complex Optimization Method Algorithm, Yapo *et al.*, 1997), optimiseur multi-objectif, qui converge vers un minimum global dans l'espace des fonctions objectif. Cette méthode requiert un nombre élevé d'itérations et demande un temps de calcul conséquent. L'étude est réalisée sur le bassin de l'Amazone, en utilisant les données ENVISAT disponibles pour l'année 2006 sur 16 stations virtuelles situées le long du fleuve.

Le projet de modélisation proposé par le CERFACS consiste donc à appliquer une nouvelle méthode d'optimisation au modèle HyMAP, moins coûteuse que MOCOM-UA : la méthode **BC-DFO** (Bound-Constrained Derivative-Free Optimization, Gratton *et al.*, 2010). Contrairement à MOCOM-UA, cet optimiseur est mono-objectif et local. Pour que cette méthode soit efficace, il faut que la fonction objectif admette un nombre réduit de minima locaux, induisant une convergence rapide vers un minimum local, que l'on espère proche du minimum global. L'objectif suivant est de tester les nouveaux résultats et de les comparer à l'optimisation établie par l'algorithme MOCOM-UA. La qualité de l'optimisation BC-DFO sur le modèle HyMAP sera ensuite discutée.

Le projet s'est aussi ouvert au domaine de l'assimilation de données. Cette technique permet de minimiser une fonction objectif d'écart entre simulations et observations en prenant en compte les erreurs de mesure liées aux instruments et les erreurs d'ébauche liées au modèle. L'assimilation de données est historiquement utilisée en météorologie car les modèles de prévisions de phénomènes naturels doivent intégrer des observations et des ébauches afin de se caler au mieux et ainsi gagner en précision de prévision.

Le rapport s'articule en cinq parties. La partie 1 pose le problème en le situant dans son contexte. Dans la partie 2, les deux méthodes d'optimisation mises en jeu dans ce projet, MOCOM-UA et BC-DFO, sont décrites en détail et comparées pour montrer l'intérêt d'appliquer une nouvelle méthode d'optimisation à HyMAP. La partie 3 expose la méthodologie employée pour faire communiquer l'algorithme BC-DFO avec HyMAP. Dans la partie 4, les résultats sont discutés en montrant l'impact de l'algorithme BC-DFO sur l'optimisation des paramètres du modèle. Enfin, la partie 5 décrit l'ouverture de ce projet à l'assimilation de données avec l'algorithme BC-DFO nouvellement fonctionnel pour la calibration de HyMAP.

Partie 1

Position du problème

1.1 Le modèle d'hydrologie à échelle globale HyMAP

HyMAP est un modèle d'hydrologie à l'échelle globale permettant de simuler le débit et la hauteur d'eau des rivières. Son principe général de fonctionnement est décrit par la figure 1.1.

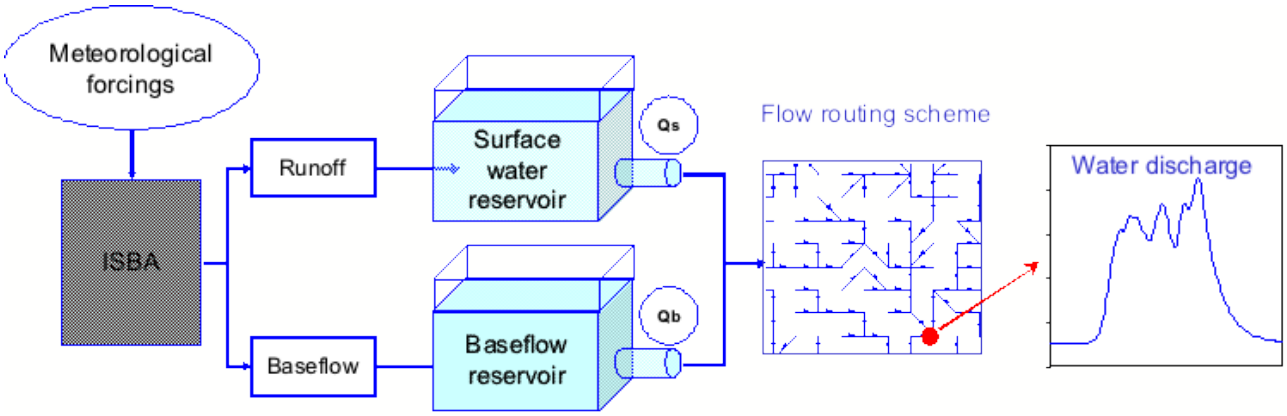


FIGURE 1.1 – Schéma du fonctionnement général du modèle HyMAP

Ce modèle est couplé à **ISBA-LSM** (Interactions Sol-Biosphère-Atmosphère - Land Surface Model, Noilhan *et al.*, 1996) qui, sous les contraintes météorologiques, lui fournit des données de ruissellement (Runoff), drainage (Baseflow) et évapotranspiration. HyMAP calcule des temps de transfert de l'eau par ruisselement (Q_s) et par drainage (Q_b) et, *via* le schéma de routage des rivières **CaMa-Flood** (Catchment-based Macro-scale Floodplain model, Yamazaki *et al.*, 2011), établit des cartes de hauteurs et de débit d'eau pour un jour donné.

Le modèle HyMAP opère à l'aide de sept paramètres spatialisés sur la grille du modèle. Ici, nous cherchons à optimiser quatre d'entre eux (Getirana *et al.*) :

- la **largeur** W et la **hauteur** H du lit de rivière,
- le **coefficient de Manning** n_r du lit de rivière (rugosité),
- le **temps de transfert par drainage** T_b de la rivière.

En pratique, on part de valeurs prédéfinies pour chaque maille du modèle, et on cherche des coefficients multiplicateurs globaux, car il serait trop coûteux de spatialiser l'optimisation. Ces coefficients multiplicateurs forment un jeu de paramètres, vecteur noté x de dimension 4 dans notre problème.

1.2 Description et utilisation des données satellites

Les données d'observation sont nécessaires pour calibrer les paramètres du modèle HyMAP. En effet, elles sont intégrées dans la formulation de la fonction objectif minimisée par les algorithmes d'optimisation.

ENVISAT (Environment Satellite), mis en orbite le 1er Mars 2002, est un satellite conçu par l'Agence Spatiale Européenne ayant pour objectif de surveiller les changements climatiques et environnementaux (Fig. 1.2). Dix instruments optiques et radars sont embarqués à son bord dans le but de fournir des données continues sur l'état de la surface terrestre, l'atmosphère, l'océan et la calotte glaciaire. Le cycle de son orbite polaire héliosynchrone (période de révolution de 101 minutes) est réalisée en trente cinq jours. Les observations de ce satellite sont utilisées dans de nombreuses disciplines des géosciences (chimie atmosphérique, océanographie, hydrologie, etc).

FIGURE 1.2 – Illustration du satellite ENVISAT (*CNES*)

L'optimisation des paramètres de HyMAP par l'optimiseur MOCOM-UA utilise les données altimétriques du satellite, dont la précision est de quelques centimètres. La qualité de ces données permet d'améliorer les paramètres et donc les simulations d'HyMAP. Seize stations altimétriques virtuelles, situées le long de l'Amazone et alimentées par les observations d'ENVISAT sur la période 2002 à 2006, sont sélectionnées.

Notons que les mesures altimétriques d'ENVISAT ont pour référence le niveau de la mer. Il y a donc un biais entre les hauteurs d'eau des rivières simulées par HyMAP avec comme référence le fond du lit. Ce biais est éliminé en comparant les différences entre les hauteurs d'eau et leurs moyennes respectives.

1.3 Description des fonctions objectif

Le coefficient **Nash-Sutcliffe** est un critère classique utilisé en hydrologie (Getirana *et al.*). Il exprime le rapport entre l'écart quadratique entre les observations et les simulations et l'écart quadratique moyen des observations et varie entre $-\infty$ et 1.

$$NS = 1 - \frac{\sum_{t=1}^{nt}(O_t - S_t)^2}{\sum_{t=1}^{nt}(O_t - \bar{O})^2},$$

où t est le temps, nt le nombre total de jours d'observation, O et S respectivement la valeur de l'observation et de la simulation, $\bar{O}$ et $\bar{S}$ les moyennes de ces valeurs pour la période étudiée (Getirana *et al.*).

Dans notre situation, ce genre de critère ne convient pas. En effet, comme mentionné précédent, un biais lié à la différence entre les références d'évaluation des hauteurs d'eau de HyMAP et ENVISAT doit être pris en compte.

Deux critères sont alors retenus pour l'optimisation. Tous deux sont fondés sur une comparaison des différences entre les hauteurs simulées par HyMAP et les observations de hauteurs d'eau du satellite ENVISAT et leurs moyennes respectives.

Ces deux critères sont formulés comme suit :

$$NSA = \frac{\sum_{t=1}^{nt}[(O_t-\bar{O})-(S_t-\bar{S})]^2}{\sum_{t=1}^{nt}(O_t-\bar{O})^2},$$

$$WR^2 = \begin{cases} |\alpha|.R^2 \text{ si } \alpha \leq 1 \\ |\alpha|^{-1}.R^2 \text{ si } \alpha > 1 \end{cases} \quad \text{avec } R^2 = 1 - \left(\frac{\sum_{t=1}^{nt}(O_t-\bar{O})(S_t-\bar{S})}{\sqrt{\sum_{t=1}^{nt}(O_t-\bar{O})^2}\sqrt{\sum_{t=1}^{nt}(S_t-\bar{S})^2}} \right)^2,$$

où R^2 est le coefficient de corrélation des niveaux d'eau et α la dérivée de la régression linéaire entre les signaux observés et les signaux simulés (Getirana *et al.*).

Ces deux critères constituent les deux **fonctions objectif** utilisées dans la méthode d'optimisation MOCOM-UA. Cette dernière a pour but de minimiser ces deux fonctions à la fois en construisant un front de Pareto (Ehrgott, 2005) à partir de différents jeux de paramètres. Dans la méthode d'optimisation mono-objectif BC-DFO, on définit la fonction coût à minimiser comme la somme de ces deux critères.

1.4 Optimisation multi- et mono-objectif

L'un des avantages de l'algorithme de minimisation multi-objectif MOCOM-UA est qu'il converge vers un **minimum global**. À partir d'une population de jeux de paramètres initiaux, la minimisation multi-objectif MOCOM-UA consiste à faire tendre la répartition de la population initiale vers un front de Pareto, c'est-à-dire l'ensemble des points ne pouvant être jugés meilleurs les uns par rapport aux autres vis-à-vis des fonctions côut. Un deuxième avantage de cette méthode est la possibilité de choix du jeu de paramètres optimal parmis les cent qui constituent le front de Pareto. Ce choix peut s'effectuer selon le sens physique des valeurs des paramètres. En pratique, le jeu de paramètre optimal est choisi de manière automatique comme le point central du front de Pareto (ce choix sera justifié par la suite). L'inconvénient de cet algorithme est le coût en temps de calcul. En effet, en partant de cent points (jeux de paramètres initiaux), la minimisation nécessite beaucoup d'évaluations de la fonction objectif. Dans le cas de HyMAP, il y a un peu plus de 4500 évaluations de fonction, ce qui dure un peu plus de neuf jours sur un ordinateur de calcul standard.

C'est pourquoi nous allons nous intéresser à un algorithme mono-objectif, BC-DFO. Ce dernier converge vers un **minimum local** cette fois-ci. Le risque que nous prenons est donc de trouver un jeu de paramètres optimal assez éloigné du minimum global. Cependant, il sera vu dans la partie 4 que ce risque peut être maîtrisé selon les conditions initiales que l'on impose à l'algorithme.

Les deux méthodes d'optimisation présentées dans cette section seront détaillées dans la deuxième partie de ce rapport.

Partie 2

Méthodes d'optimisation : MOCOM-UA et BC-DFO

2.1 Introduction

Dans le contexte de la calibration de paramètres, il est nécessaire d'avoir recours à des optimiseurs. L'optimisation revient à minimiser une fonction coût f, constituée d'une ou plusieurs fonction(s) objectif, comparant ici les simulations et les observations (méthode mono- ou multi-objectif). Cette minimisation se fait généralement à l'aide du gradient de cette fonction, ∇f, qui permet notamment de construire un modèle quadratique m. Si la fonction coût est trop complexe, il devient coûteux de calculer son gradient. D'autres moyens doivent donc être employés afin, par exemple, de construire le modèle m par interpolation. Pour respecter les limites imposées par la physique, il est nécessaire de contraindre la minimisation à l'aide de bornes. En effet, nous devons éviter autant que possible les jeux de paramètres n'ayant pas de sens physique. Pour plus d'informations sur l'optimisation, le lecteur pourra se référer à Nocedal *et al.*, 1999.

2.2 Description de l'optimiseur MOCOM-UA

MOCOM-UA est l'optimiseur actuellement utilisé avec HyMAP. Ce n'est pas un optimiseur classique, tout d'abord parce qu'il est multi-critères, et ensuite parce qu'il n'utilise pas le gradient de la fonction à minimiser. C'est un algorithme évolutionnaire (Yapo *et al.*, 1997).

2.2.1 Principe général

Le processus d'optimisation avec MOCOM-UA part d'une population de cent jeux de paramètres aléatoires, ce nombre étant considéré comme suffisant pour garantir la convergence pour notre problème. Le principe général consiste alors à faire évoluer les plus mauvais jeux de paramètres en utilisant de meilleurs jeux, quitte à dépasser ces derniers qui seront alors eux-mêmes remis en question. Par itérations successives, qui nécessitent des évaluations des fonctions objectif, on arrive à une solution pour laquelle aucun jeu de paramètres n'est considéré comme moins bon que les autres, c'est le front de Pareto (Ehrgott, 2005).

2.2.2 Méthode

Chaque jeu de paramètres peut être représenté comme un point dans l'espace des paramètres, illustré par la figure 2.2a dans le cas de deux paramètres. On peut alors calculer les fonctions coût correspondantes après une simulation réalisée à partir de ce jeu de paramètres. On représente

ainsi le point (jeu de paramètres) dans l'espace des fonctions coût (Fig. 2.2b). L'optimisation consistant à minimiser ces fonctions, on cherche à faire tendre les points vers les minima de l'espace des fonctions, en déplaçant les jeux de paramètres dans leur propre espace (par des réflections ou des contractions). Il faut effectuer une nouvelle simulation HyMAP à chaque itération pour ré-évaluer les fonctions coût et ainsi savoir si le point a été amélioré dans l'espace de ces fonctions.

2.2.3 Description de l'algorithme

Tel que le processus d'optimisation est décrit dans la figure 2.1, on choisit cent points de façon aléatoire dans le domaine des paramètres autorisés. Puis on lance une simulation de HyMAP avec chacun des jeux de paramètres correspondants (ETAPE 1 de la figure 2.1). À l'issue de cette première simulation, l'algorithme effectue un classement de chaque point (ETAPE 2 de la figure 2.1) selon qu'il soit dominé ou non par les autres. La domination d'un point y_2 est définie mathématiquement par le fait qu'il existe y_1 tel que :

$$\begin{array}{l}\text{pour tout } \ell = 1, .., p,\ [f(y_1)]_\ell \leq [f(y_2)]_\ell \text{ et} \\ \text{il existe } \ell \in \{1, .., p\},\ [f(y_1)]_\ell < [f(y_2)]_\ell.\end{array}$$

Alors on dit que y_1 *domine* y_2 et $f(y_1)$ *domine* $f(y_2)$. Nous pouvons désormais formellement définir ce qu'est une solution optimale de *Pareto* y^* du problème d'optimisation multi-objectif :

$$\begin{array}{l} y^* \text{ est une solution optimale de } \textit{Pareto} \\ \qquad \Leftrightarrow \quad \nexists\, y \text{ tel que } \left\{ \begin{array}{l} \text{pour tout } \ell = 1, .., p,\ [f(y)]_\ell \leq [f(y^*)]_\ell \text{ et} \\ \text{il existe } \ell \in \{1, .., p\},\ f_\ell(y) < f_\ell(y^*), \end{array} \right. \end{array}$$

c'est-à-dire qu'il n'existe aucun point y qui domine y^*.

Le classement de ces points s'effectue comme suit :

1. les points qui ne sont dominés par aucun autre point : ils constituent le front de Pareto courant. Un point domine un autre point lorsqu'il donne un meilleur résultat sur chacun des critères d'optimisation (fonctions coût).
2. Les points qui sont dominés seulement par les points appartenant au front de Pareto. Ils constituent le deuxième front.
3. Les points qui sont dominés seulement par les points appartenant aux deux premiers fronts. Ils constituent le troisième front*etc.*

Le dernier front regroupe les points qui n'en dominent aucun autre. Ces points sont les moins pertinents. Par la suite, ils seront appelés "mauvais points". Ils feront donc l'objet d'une amélioration avant la prochaine itération. Pour cela, on choisit de manière aléatoire $n + 1$ points pour définir un simplexe [1] et pour en calculer le barycentre. Ce dernier servira de point central pour déterminer les points de réflection ou de contraction qui correspondront alors à l'amélioration apportée.

On construit un point de réflection, considéré comme candidat "nouveau point" (Fig. 2.2a). Pour être conservé, le point de réflection doit respecter deux exigences :

– être dans la zone autorisée initialement (Fig. 2.2a),
– ne pas être dominé par au moins un point du simplexe (Fig. 2.2b).

Sur la figure 2.2, on voit que le point de réflection respecte les critères énoncés précédemment.

1. Un simplexe est un ensemble de n+1 jeux de paramètres, dont le jeu à améliorer, n étant la dimension du problème (ici, n=4).

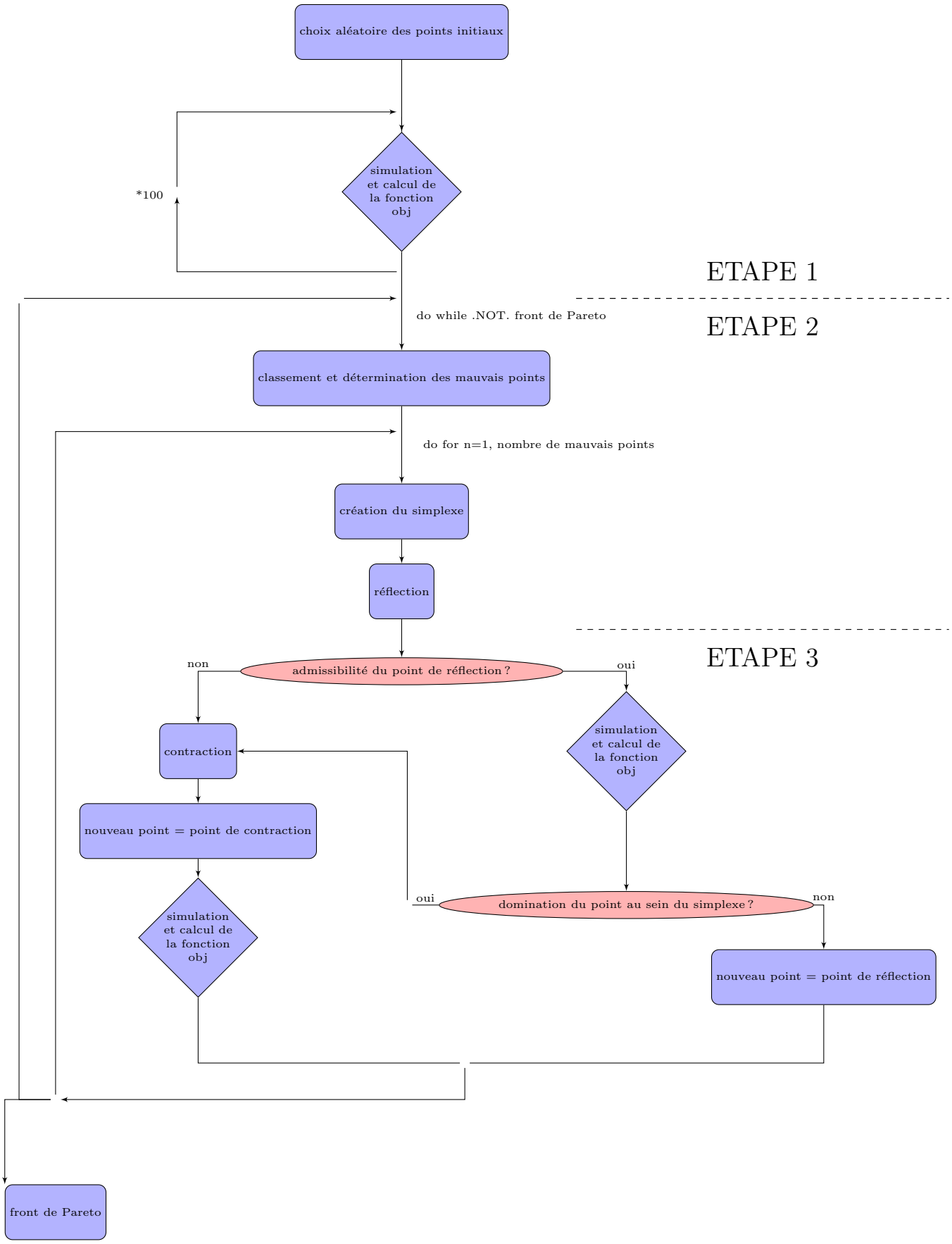
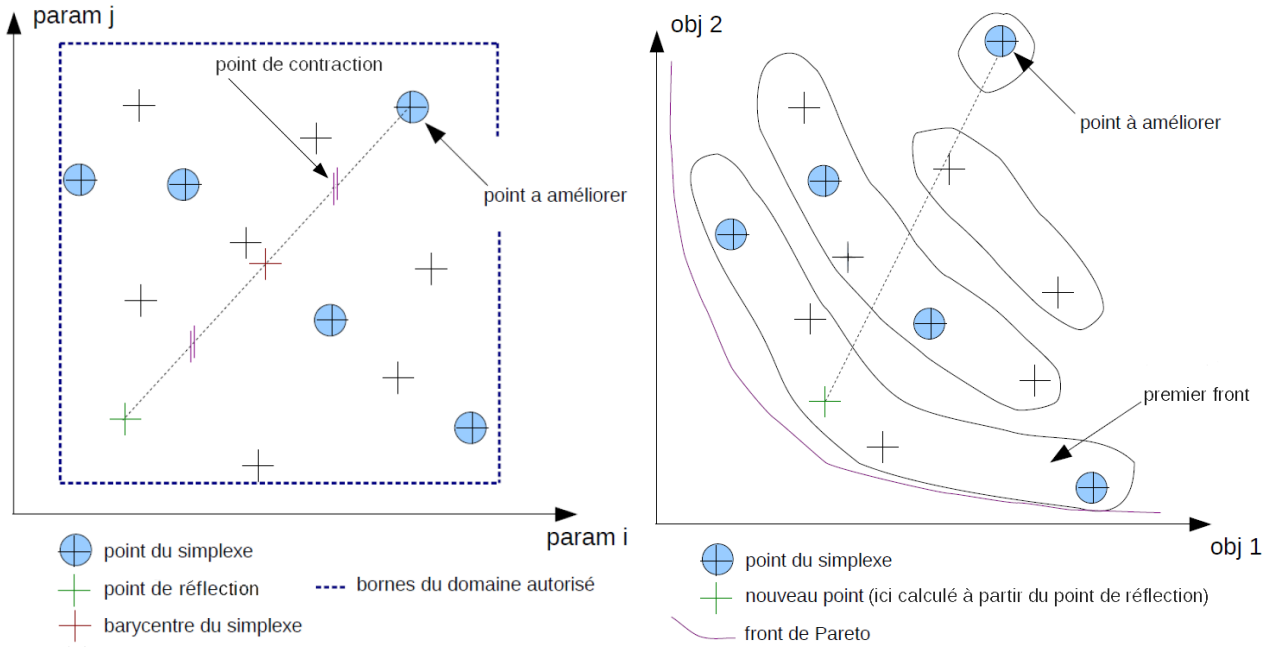


FIGURE 2.1 – Schéma de l'algorithme MOCOM-UA



(a) Espace des paramètres

(b) Espace des fonctions objectif

FIGURE 2.2 – Schéma de la minimisation MOCOM-UA

Pour vérifier la deuxième condition, il faut évaluer la fonction objectif et donc faire tourner le modèle avec les paramètres correspondant au nouveau point. Si une de ces deux conditions n'est pas respectée, on choisit le point situé à la moitié du chemin entre le barycentre du simplexe et le point à déplacer (contraction), et on relance une simulation avec les paramètres correspondants (ETAPE 3 de la figure 2.1).

Une fois que tous les mauvais points ont été améliorés, on trie de nouveau tous les points en fronts successifs et on recommence le processus d'amélioration des nouveaux mauvais points. Une itération a donc pour but d'améliorer tous les mauvais points de l'ensemble. Elle requiert autant d'évaluations de fonctions coûts que le nombre de mauvais points, voire plus en cas de contraction.

Si cet optimiseur converge toujours vers le front optimal, il a pour défaut de lancer énormément de simulations du modèle. Pour la calibration de HyMAP, MOCOM-UA requiert plus d'un millier d'itérations pour converger, et plus de 4500 évaluations de fonctions coût.

2.3 Description de l'optimiseur BC-DFO

La méthode d'optimisation BC-DFO (Bound-Constrained Derivative-Free Optimization, Gratton *et al.*, 2010) est une méthode mono-objectif locale sans dérivées par région de confiance avec contraintes de bornes. Les deux fonctions objectif d'HyMAP sont donc résumées en une seule par simple sommation, ceci étant permis à la fois par la forme convexe et connexe du front de Pareto établi par MOCOM-UA et par le fait de ne choisir qu'un seul point sur le front (comme expliqué dans le paragraphe 1.1.4).

Pour comprendre le fonctionnement de cette méthode, le principe de l'optimisation par région de confiance sans contrainte et avec gradient sera expliqué en premier lieu. Enfin, il sera question de l'optimisation sans gradient, puis avec contraintes de bornes.

L'optimisation avec contraintes est la minimisation d'une fonction objectif f qui respecte des conditions, imposées dans notre cas par la physique. Ces contraintes s'appliquent ici aux paramètres.

On peut imposer ces contraintes par l'intermédiaire de fonctions, par exemple g et h telles qu'un paramètre x respecte $g(x) \leq 0$ et $h(x) = 0$. Dans le cas le plus simple, on applique directement des bornes au paramètre x : $l_b \leq x \leq u_b$ avec l_b et u_b les bornes inférieure et supérieure respectivement. Ce dernier type de contrainte, appelé contrainte de bornes, est traité dans la méthode d'optimisation BC-DFO.

2.3.1 Optimisation par région de confiance

L'optimisation par région de confiance consiste à minimiser un modèle quadratique m dans un espace restreint qui évolue itérativement. Si le gradient de la fonction objectif est connu, nous pouvons directement construire le modèle quadratique en utilisant le développement de Taylor à l'ordre deux. Cependant, la fonction objectif d'HyMAP est trop complexe pour nous permettre de calculer son gradient par rapport aux paramètres à optimiser. On peut contourner ce problème en construisant un modèle quadratique par interpolation par exemple.

Cas où le gradient de la fonction est connu

L'algorithme d'optimisation peut être décrit en cinq étapes répétées à chaque itération et jusqu'à convergence :

1. On construit le modèle quadratique $m_k(x_k)$ à partir du gradient de la fonction objectif. On peut toujours définir une région de confiance assez restreinte au sein de laquelle le développement de Taylor à l'ordre 2 est valide. Pour un pas s, on a :
$$m_k(x_k + s) = f(x_k) + \nabla f(x_k)^t s + \frac{s^t \nabla^2 f(x_k) s}{2}.$$
2. On exprime la région de confiance ainsi : $B_k = B(x_k, \Delta_k)$, boule de centre x_k et de rayon Δ_k. Comme son nom l'indique, la région de confiance est une zone dans laquelle on peut faire confiance au modèle m_k. Plus le modèle sera valide, plus la région de confiance s'étendra sur un grand espace.
3. On minimise le modèle à l'aide du gradient conjugué et on obtient une valeur s_k pour le pas. On pose alors : $x_k^+ = x_k + s_k$
4. On calcule le ratio entre la réduction obtenue et celle prédite par le modèle :
$$\rho_k = \frac{f(x_k) - f(x_k^+)}{m_k(x_k) - m_k(x_k^+)}.$$
Le ratio ρ_k est petit si la réduction du modèle est grande par rapport à la réduction de la fonction objectif. Cela signifie qu'on ne peut faire confiance au modèle quadratique pour prédire la décroissance dans la zone de confiance et qu'il faut modifier le rayon Δ_k afin de la restreindre.
À l'opposé, ρ_k se rapproche de 1 si la réduction du modèle est comparable voire inférieure à la réduction de la fonction objectif. Cela signifie que le modèle quadratique est une bonne approximation de la fonction coût au sein de la région de confiance.
5. On accepte donc le pas s_k ou non suivant la valeur de ρ_k :

   $\rho_k < \eta_1$: rejet de $s_k \Rightarrow x_{k+1} = x_k$ et $\Delta_{k+1} = \frac{\Delta_k}{2}$,

   $\rho_k > \eta_1$: acceptation de $s_k \Rightarrow x_{k+1} = x_k^+$,

   $\rho_k > \eta_2 \Rightarrow \Delta_{k+1} = \Delta_k * 2$.

   Des choix classiques pour ces constantes sont $\eta_1 = 0.1$ et $\eta_2 = 0.9$.

Cas où le gradient de la fonction est inconnu

L'algorithme d'optimisation est comparable au précédent, mais nous devons utiliser une technique particulière pour construire le modèle quadratique car il est impossible de calculer le gradient de la fonction objectif. Au lieu d'un développement de Taylor à l'ordre 2, le modèle quadratique est calculé par une interpolation polynomiale p :

$$m(x+s) = p(x+s) = \sum_{i=1}^{p} \alpha_i \Phi_i(x+s),$$

où les α_i représentent les coefficients que l'on affecte aux éléments $\Phi_i(x)$ de la base $\{\Phi_i\}_{i=1}^{p}$ des polynômes quadratiques.

La base $\{\Phi_i\}_{i=1}^{p}$ peut être choisie de différentes manières :

- polynômes de Lagrange (cas de la méthode BC-DFO)
- polynômes de Newton
- base naturelle des polynômes (ex : $n = 2 \Rightarrow \Phi = \{1, x, y, xy, x^2, y^2\}$ où n est le nombre de paramètres).

Les coefficients α_i sont solutions du système linéaire suivant :

$$M(\Phi, Y)\alpha = f(Y) \text{ où } \begin{cases} f(Y) \in \mathbb{R}^p \text{ et } [f(Y)]_i = f(y^i), \\ \alpha \in \mathbb{R}^p \text{ et } [\alpha]_i = \alpha_i, \\ M(\Phi, Y) \in \mathbb{R}^{p*p} \text{ et } [M(\Phi, Y)]_{i,j} = \Phi_j(y^i), \end{cases}$$

où $Y = \{y^i\}$ est un ensemble d'échantillonnage de p points

Pour résoudre le système linéaire précédent, il faut évaluer la fonction f en chacun des points de l'échantillonnage. Or l'échantillonnage se fait avec $p = \frac{(n+1)*(n+2)}{2}$ points, où n est le nombre de paramètres à optimiser, de façon à pouvoir représenter complètement la base des polynômes quadratiques. Dans notre cas, $n = 4$ et $p = 15$. Cependant, à l'initialisation du processus d'optimisation BC-DFO, l'échantillonnage se fait à partir de seulement $n + 1$ points, ce qui sous-conditionne le problème de résolution du système linéaire faisant intervenir M, α et f. Cela permet toutefois d'économiser des évaluations de fonction coût. Le nombre de points échantillonnés augmente progressivement pour atteindre p. À chaque nouvelle itération, le modèle est reconstruit à l'aide des p points les plus proches du point courant en prenant en compte tous les points dont la fonction objectif a été évaluée.

2.3.2 Optimisation avec contraintes de bornes

Lorque les minima se situent sur les bornes, on dit alors que les contraintes sont actives à la solution. Si on applique une méthode de région de confiance sans dérivée en traitant les bornes simplement, un problème apparaît lors de la minimisation. En effet, dans l'espace des paramètres, si nous atteignons une borne dans l'une des dimensions, la fonction peut évoluer avec une dimension fixée à la valeur de la borne atteinte. Dans ce cas précis, la matrice d'interpolation M est alors mal conditionnée, ce qui rend le modèle quadratique particulièrement mauvais.

Pour contourner ce problème, nous devons fixer la dimension concernée à la valeur de la borne, puis minimiser dans le sous-espace des dimensions n'ayant pas encore atteint leurs bornes.

2.4 Illustration de la minimisation avec BC-DFO

La figure 2.3 montre un exemple de minimisation d'une fonction objectif, représentée par la surface rouge, par l'algorithme BC-DFO sur un problème à deux variables ($n = 2$). Une première interpolation à partir de trois points initiaux permet de construire le modèle m représenté par la surface bleue. La minimisation de ce modèle quadratique donne un nouvel itéré pour lequel on évalue la fonction objectif. On reconstruit alors le modèle quadratique à partir des quatres points (la figure 2.3a représente cette même étape avec cinq points et un point minimiseur). Une fois les six points obtenus, le point le plus éloigné de l'itéré courant est remplacé par le point nouvellement obtenu.(Fig 2.3b). L'objectif final est d'obtenir un modèle m qui soit le plus proche possible de la forme de la fonction objectif f sur la région de confiance.

2.5 Discussion sur la méthode BC-DFO

Un des obstacles majeurs de l'utilisation d'un optimiseur tel que BCDFO est la possible existence de minima locaux distincts du minimum global. À titre illustratif, afin de nous faire une idée de la forme des deux fonctions objectif à minimiser, NSA et WR^2, nous avons decidé de les représenter dans le sous-espace des paramètres T_b et η_r, les deux autres, W et H, étant fixés arbitrairement à 1. Cette étude n'est pas transposable lorsqu'on augmente le nombre de paramètres et lorsqu'on change certains de ces paramètres.

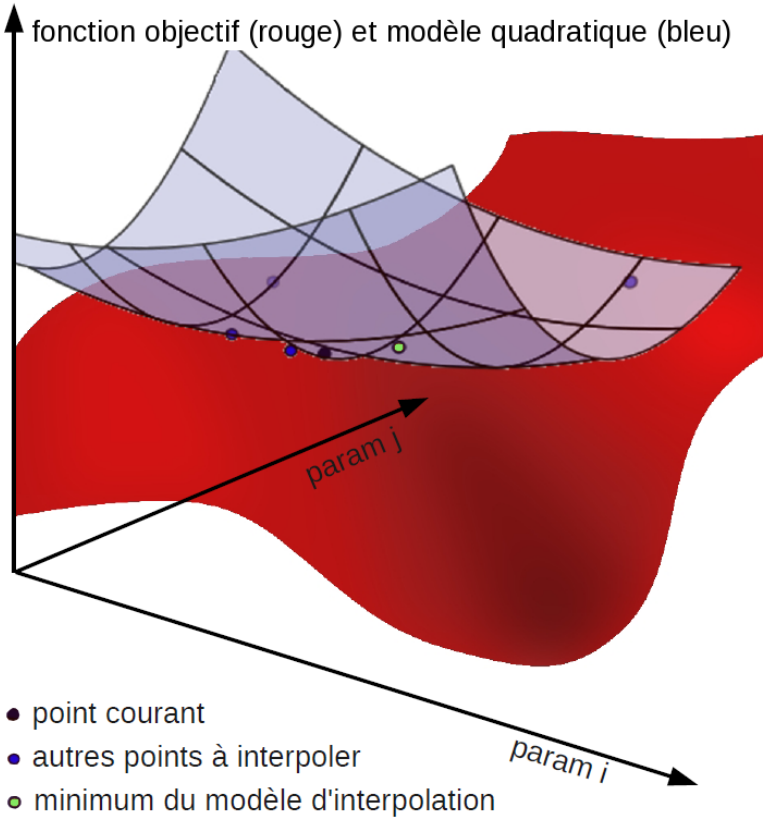
Les figures 2.4a et 2.4b représentent les surfaces qui en ont été déduites en faisant tourner HyMAP sur un maillage rectangulaire.

On remarque tout d'abord qu'il existe selon toute vraisemblance un ensemble convexe contenant les minima de NSA et WR^2 dans laquelle ces deux fonctions sont convexes. Si tel est bien le cas, alors la fonction somme est aussi convexe dans l'ensemble. On a donc équivalence entre minimum local et global. Si on a bien - comme semble l'indiquer la figure 2.4 - deux fonctions objectifs convexes dans une même zone contenant leurs minima, le problème de l'existence de minima locaux ne se pose plus. L'hypothèse de "proximité" des minima est probable, notamment grâce au fait que les deux fonctions objectif utilisent les mêmes observations mais leurs formules diffèrent.

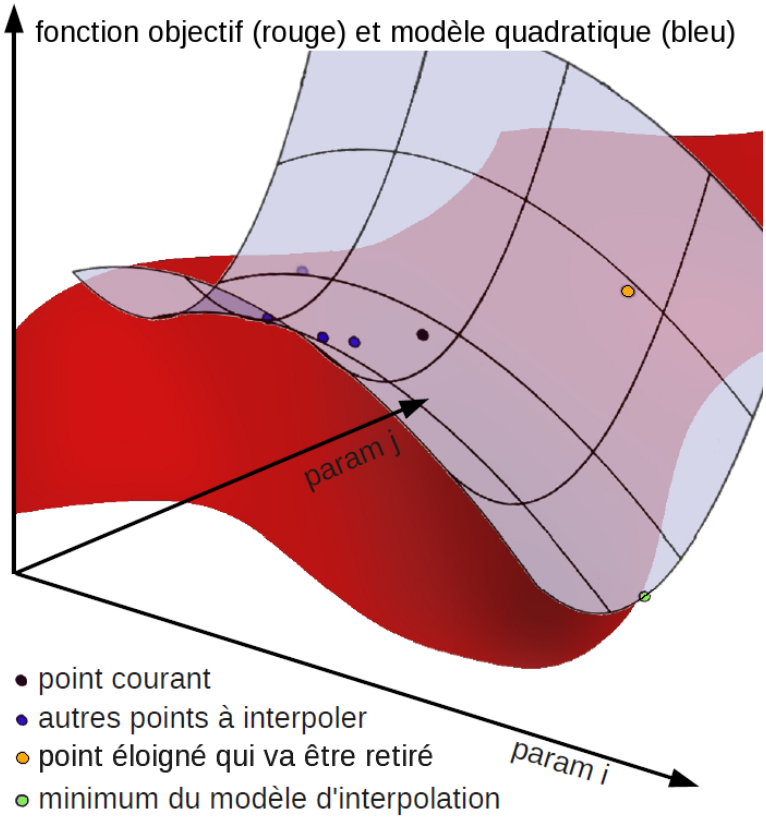
Pour affiner le résultat, nous avons effectué des zooms successifs avec des maillages plus serrés, sans que les fonctions ne semblent rompre avec la propriété de convexité.

On peut néanmoins reprocher à la méthode faisant appel à la somme de favoriser la fonction qui présente la plus forte convexité. En effet, en supposant que nos fonctions soient convexes, leur gradient croit à mesure que l'on s'éloigne du minimum où il est nul. Or le minimum de la somme $NRA + WR^2$ est donné par le point qui annule la somme des deux gradients. Si, par exemple, NSA a un gradient qui croit plus vite que WR^2, l'annulation du gradient de leur somme se fera plus près du point qui minimise NSA comme le montre la figure 2.5.

Il apparaît alors intéressant de pondérer la somme des fonctions coût. Cela pourrait être étudié en suivant par exemple la méthode de Logist *et al.*, 2011. En effet, il paraît naturel d'optimiser équitablement les deux fonctions coûts. La pondération de la somme des deux fonctions coûts peut se baser sur des coefficients liés à la convexité de la fonction. Plus la fonction est convexe, plus elle attire le minimum et donc plus le coefficient de pondération doit être petit.

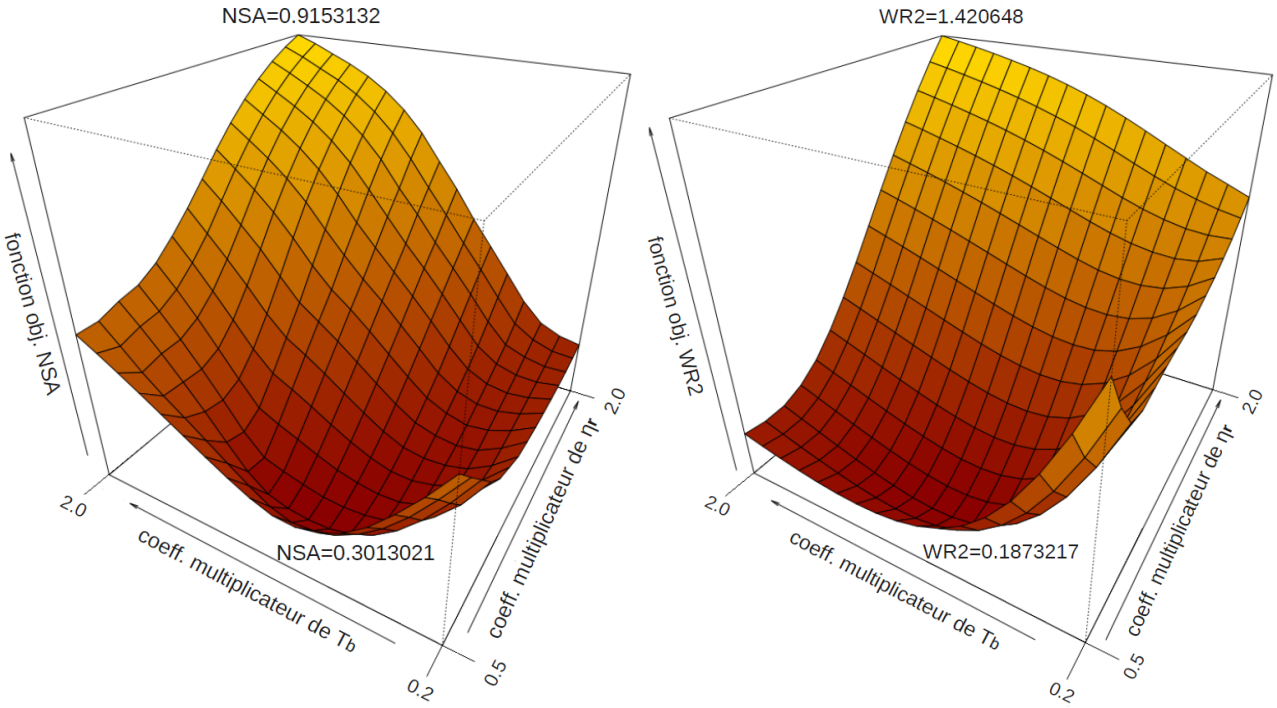


(a)



(b)

FIGURE 2.3 – Schéma de la minimisation BC-DFO



(a) (b)

FIGURE 2.4 – Représentations des fonctions objectif sur un maillage rectangulaire

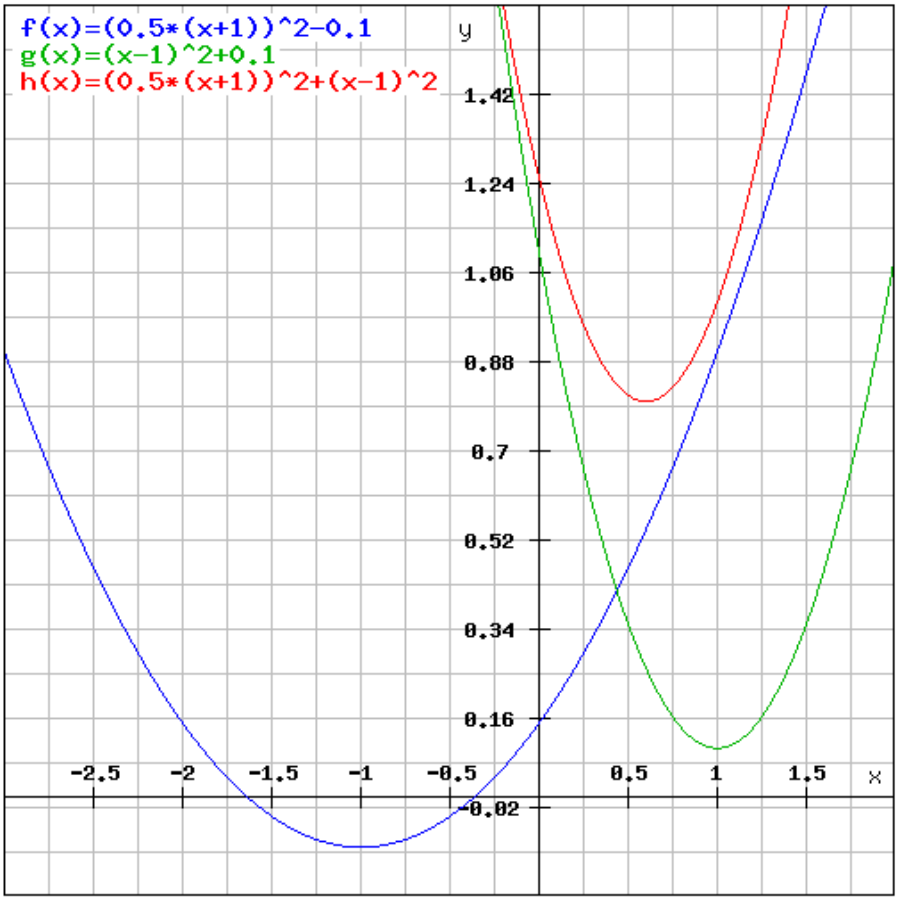


FIGURE 2.5 – Somme de deux fonctions de convexités différentes. La fonction la plus convexe a plus d'influence sur la position du minimum.

Partie 3

Méthodologie employée pour l'intégration de la méthode BC-DFO dans HyMAP

Lors de ce projet, nous avons cherché à faire communiquer l'algorithme BC-DFO avec le modèle HyMAP pour la calibration des paramètres et l'assimilation de données. Afin de bien dissocier leurs codes respectifs, des techniques de programmation ont été envisagées pour assurer la transmission des données entre eux. Le couplage de codes par PALM ou `mex-file` et la technique de communication par fichiers d'entrée/sortie ont été retenus. Cependant, la méthode PALM aurait nécessité une formation spécialisée, ne convenant pas avec le temps imparti pour ce projet.

L'algorithme d'optimisation BC-DFO est écrit en langage `Matlab` tandis que tous les modules du modèle HyMAP sont écrits en `Fortran90`. La principale difficulté est donc de faire intéragir des codes écrits dans des langages différents. L'objectif n'étant pas de réécrire les codes `matlab` en `Fortran90`, il est nécessaire de trouver un moyen de communication entre ces deux langages. L'utilisation de l'*opensource* `Octave` a été préférée à celle du logiciel `Matlab`.

Deux approches ont été envisagées durant ce projet. La première vise à coder un fichier `mex-file` pour faire intéragir la méthode d'optimisation écrite sous `Octave` avec le code HyMAP écrit en `Fortran90`. Pour plus d'informations sur le `mex-file`, le lecteur pourra se réferer à la documentation en ligne :

`http://www.mathworks.fr/support/tech-notes/1600/1605.html`.

La seconde approche, plus simple à mettre en œuvre mais plus coûteuse, se contente d'échanger des fichiers d'entrée/sortie entre les programmes `Matlab` et `Fortran`.

3.1 Structure initiale du modèle HyMAP

La figure 3.1 contient le schéma du modèle HyMAP. On peut y remarquer l'articulation entre les principales routines du code. Le modèle HyMAP peut être lancé sous deux modes principaux :

- simulation : à partir d'un jeu de paramètres, le modèle simule le débit et les hauteurs d'eau. Les fichiers de sortie contiennent les champs spatialisés de débit et de hauteurs d'eau des rivières pour tous les pas de temps.
- calibration : il s'agit de calibrer les paramètres du modèle. C'est le mode qui nous permet d'optimiser les paramètres grâce à l'algorithme MOCOM-UA.

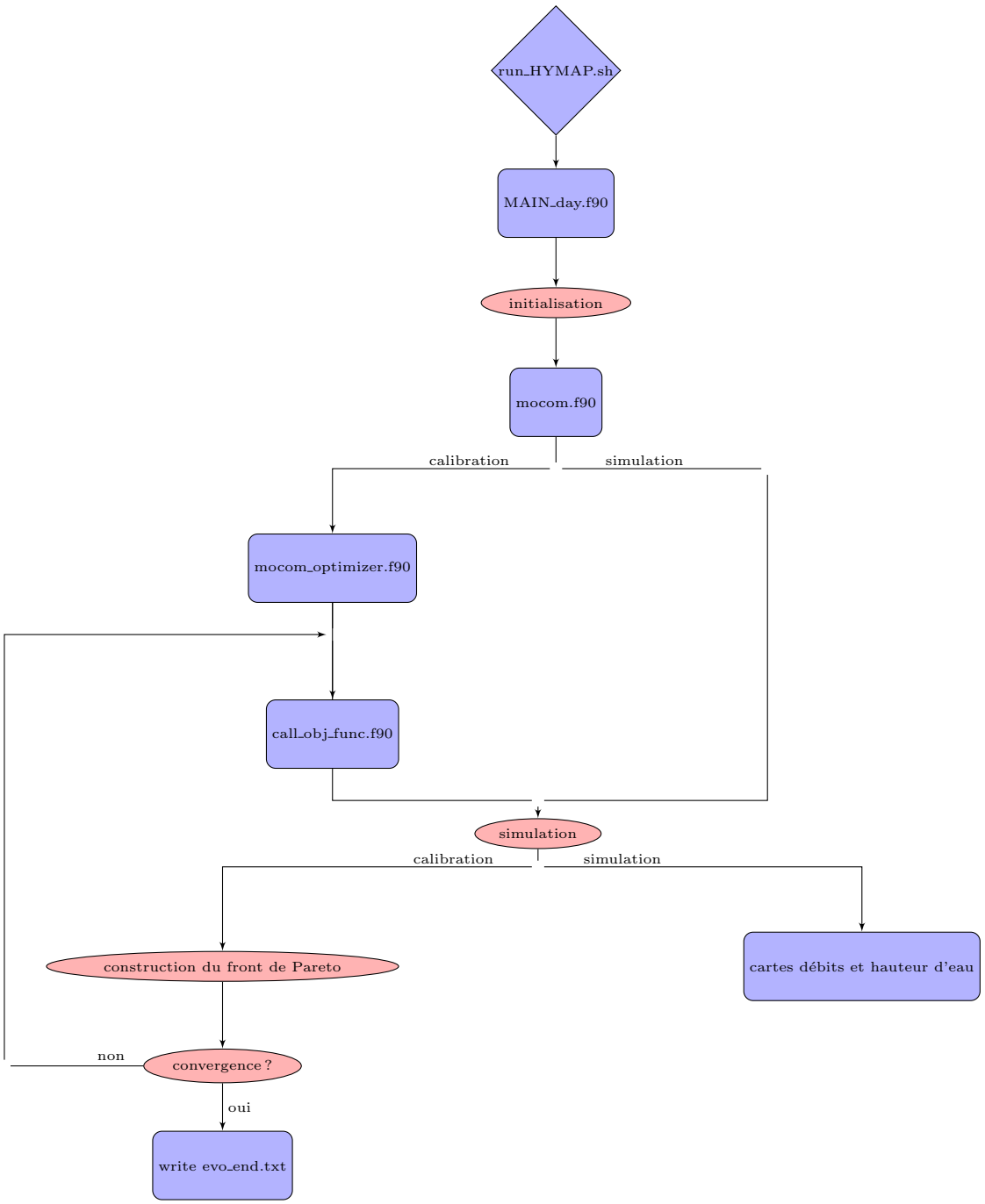


FIGURE 3.1 – Schéma de la structure initiale du modèle HyMAP

Le fichier `run_HYMAP.sh` est le lanceur du modèle. Il contient le nom du dossier où seront stockés les fichiers de sortie, le choix du mode de lancement, *etc.* Il est constitué de commandes permettant de compiler et exécuter les fichiers sources codés en `Fortran90`. Les fichiers `mod_main.f90` et `mod_calib.f90` contiennent des déclarations et initialisations de variables qui seront utilisées dans tous les autres fichiers source du modèle. Le fichier de configuration `mod_calib.f90` permet à l'utilisateur de régler de nombreux paramètres pour l'optimisation telles que la population de départ, les bornes, les fonctions objectif, *etc.*

Après le lancement de la compilation par `run_HYMAP.sh`, l'exécution du programme `MAIN_day` établit l'initialisation du modèle (paramètres). En mode calibration, au sein de la fonction `mocom.f90`, le module `mocom_optimizer.f90` est appelé : c'est dans ce dernier module que l'optimisation de la fonction objectif a lieu. Le fichier `call_obj_func.f90` coordonne les actions entre mise à jour des paramètres (*via* `update_params.f90`), simulation par HyMAP (*via*

time_loop_calib.f90), calcul des coefficients de performance pour les débits et les hauteurs d'eau (*via* call_c.f90) et calcul des deux fonctions objectif (*via* obj_func.f90). Le mode calibration intéragit donc avec le mode simulation. En effet, l'évaluation des fonctions objectif requiert à chaque itération la simulation sur deux ans des hauteurs d'eau et des débits à partir de chaque jeu de paramètres.

3.2 Modifications opérées dans HyMAP et BC-DFO par le procédé de type mex-file

La figure 3.2 représente les modifications apportées à la structure générale des codes.

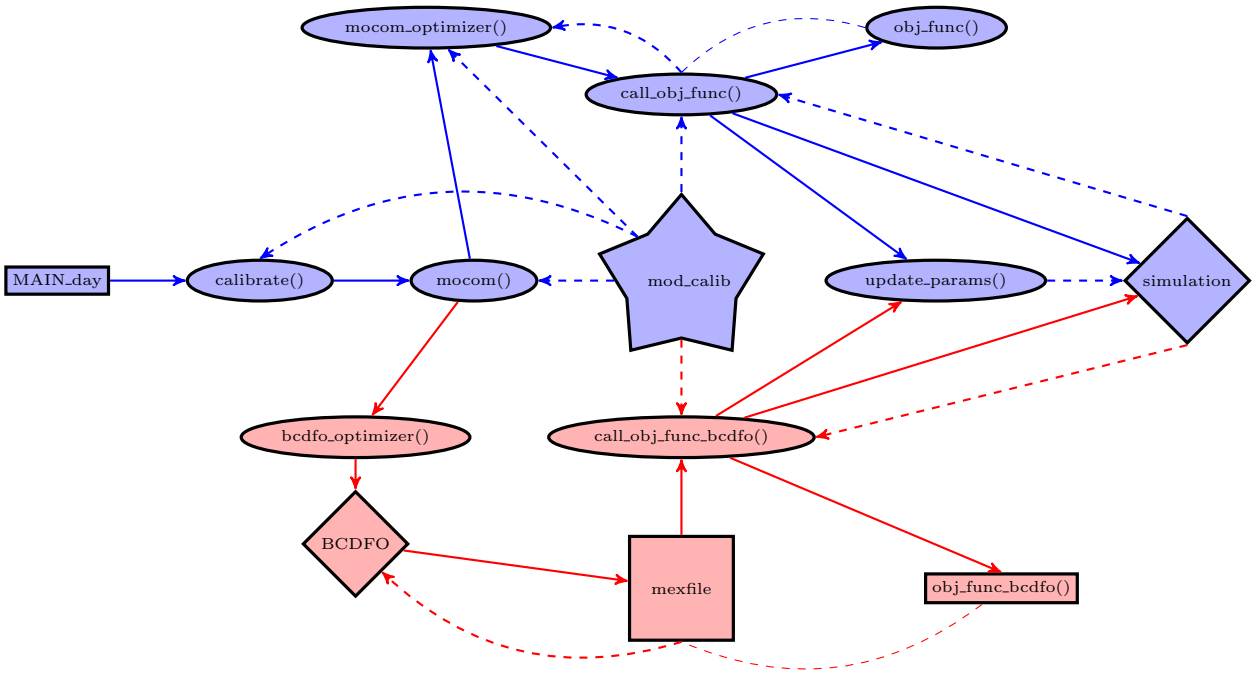


FIGURE 3.2 – Schéma de l'optimisation des paramètres par MOCOM-UA (option mex-file). Flèche pleine≡appel d'une fonction ou d'un programme ; flèche tiretée≡transfert d'informations. Description des formes : rectangle≡exécutable ; ellipse≡fonction ; losange≡programme ; carré≡mex-file ; étoile≡module de calibration

Les changements mis en place portent tout d'abord sur des fichiers pré-existants : pour rendre possible le choix de l'optimiseur (MOCOM-UA ou BC-DFO), une nouvelle variable de choix est intégrée dans mod_calib.f90. La sous-routine mocom.f90 tient alors compte de ce choix pour lancer la bonne sous-routine d'optimisation.

Il a donc été nécessaire de créer de nouvelles sous-routines telles que bcdfo_optimizer.f90 : cette sous-routine très simple appelle Octave (ou Matlab) et lui fait exécuter BC-DFO ; call_obj_func_bcdfo.f90, sous-routine presque identique à call_obj_func.f90 mis à part le fait qu'elle appelle obj_func_bcdfo.f90 et non obj_func.f90 ; obj_func_bcdfo.f90, comparable à obj_ func.f90 (voir la section 3.1), la différence étant qu'elle ne fournit pas deux mais une seule fonction objectif qui est la somme des deux autres.

Dans BC-DFO, le fichier objf.m, codé en Matlab, contient un grand nombre de fonctions de coût possibles. Nous y avons donc ajouté la fonction coût qui appelle la fonction call_obj_func_bcdfo.f90. Cet appel est assuré par le fichier mex-file nouvellement codé. Ce dernier est un fichier permettant l'exécution par Octave d'une routine Fortran, call_obj_

`func_bcdfo.f90`. Il est écrit en `C` et doit assurer le transfert des données entre `Octave` et la sous-routine.

3.3 Le fichier `mex-file`

La structure du fichier `mex-file` que nous avons codé est la suivante :

```
1  #include <config.h>
2  #include <mex.h>
3  #include <f77-fcn.h>
4
5  extern void F77_FUNC (call_obj_func_bcdfo, CALL_OBJ_FUNC_BCDFO) (double *x, double *u, double *v, double *f);
6  void mexFunction (int nlhs, mxArray* plhs[], int nrhs, const mxArray* prhs[]) {
7
8          double ff;
9          double *uu;
10         double *vv;
11         double *gg;
12         double *xx;
13
14         plhs[0]=mxCreateDoubleMatrix(13,14,mxREAL);
15         plhs[1]=mxCreateDoubleMatrix(16,14,mxREAL);
16         plhs[2]=mxCreateDoubleMatrix(1,1,mxREAL);
17
18         xx=mxGetPr(prhs[0]);
19         uu=mxGetPr(plhs[0]);
20         vv=mxGetPr(plhs[1]);
21         gg=mxGetPr(plhs[2]);
22
23         F77_FUNC (call_obj_func_bcdfo, CALL_OBJ_FUNC_BCDFO) ((double *)xx, uu, vv, &ff);
24
25         *gg=ff;}
```

Les trois premières lignes concernent le chargement de trois librairies permettant l'appel de certaines fonctions propres aux `mex-file`. La ligne suivante est la déclaration de la sous-routine codée en `Fortran90` qui est liée au fichier `objf.m` codé en `Matlab`. Le reste du code constitue la construction de la fonction `mex` elle-même. Celle-ci prend quatre arguments : les deux premiers concernent les sorties alors que les suivants sont associés aux entrées. L'appel à la sous-routine `call_obj_func_bcdfo()` se fait à la ligne 23 du code.

Le principal inconvénient de cette méthode est qu'il faudrait initialiser tous les paramètres de HyMAP à chaque appel de la sous-routine `call_obj_func_bcdfo()` car chacun de ces appels par le fichier `mex-file` constitue un nouveau processus. Plus de détails seront donnés dans la section 3.5. Pour éviter une complexification trop importante du code d'origine, nous avons décidé d'employer une autre méthode visant à échanger les entrées/sorties entre `Octave` et HyMAP par simples écriture/lecture dans des fichiers.

3.4 Modifications opérées dans HyMAP et BC-DFO par le procédé d'écriture/lecture de fichiers d'entrée/-sortie

La figure 3.3 représente les modifications apportées à la structure générale des codes. La différence structurelle entre les deux approches réside dans le fait que dans la version faisant appel aux fichiers d'entrée/sortie, une boucle de lancement s'opère entre `run_HYMAP.sh` et `run_bcdfo.m` par l'intermédiaire de `bcdfo_optimizer.f90`. L'appel au lancement de `run_bcdfo` s'effectue depuis `bcdfo_optimizer.f90` *via* la commande `Fortran` `call system(octave run_bcdfo.m)`. Ensuite, à chaque évaluation de la fonction objectif par l'algorithme d'optimisation BC-DFO, un lancement du modèle HyMAP ainsi que son initialisation sont effectués. Le transfert des données (vecteur contenant les quatre paramètres et l'évaluation de la fonction objectif) entre le modèle et `bcdfo` se fait par écriture/lecture de fichiers d'entrée/sortie nommés `param.dat` et `func_obj.dat`. Cette méthode plus fonctionnelle est donc celle que nous avons

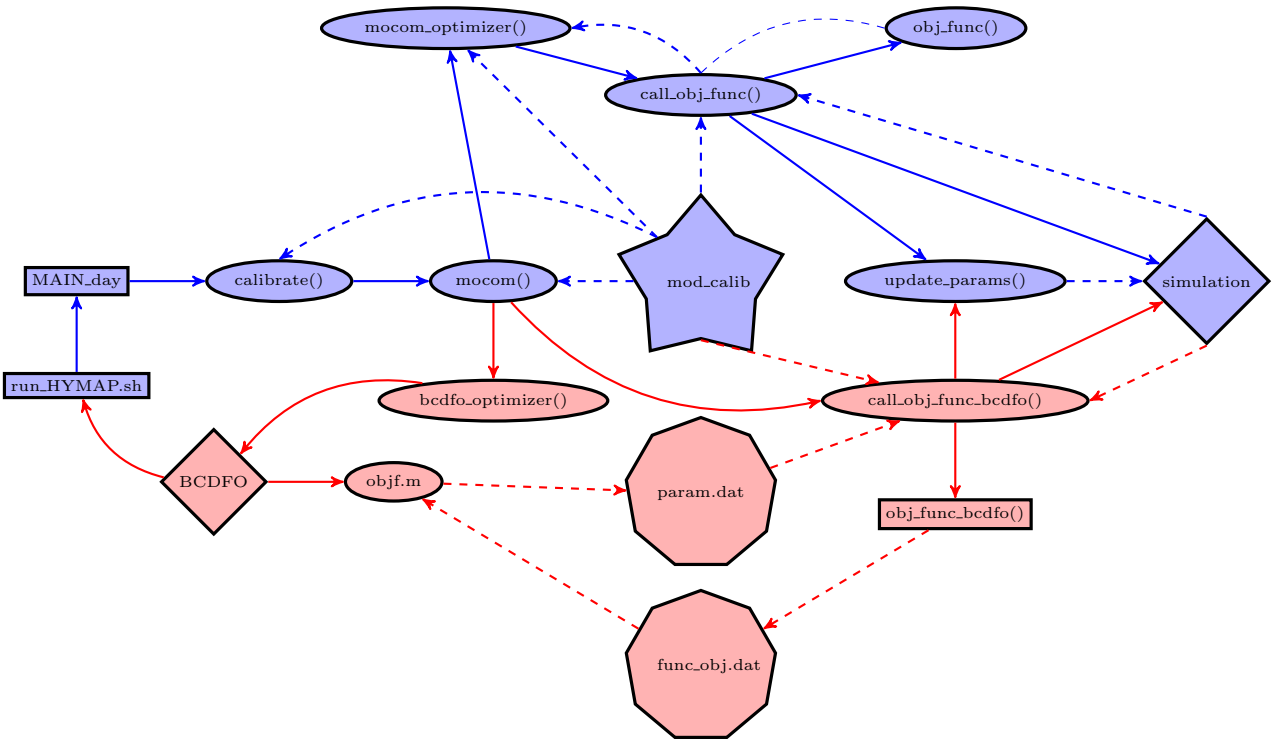


FIGURE 3.3 – Schéma de l'optimisation des paramètres par BC-DFO (écriture/lecture dans des fichiers d'entrée/sortie). Flèche pleine≡appel d'une fonction ou d'un programme ; flèche tiretée≡transfert d'informations. Description des formes : rectangle≡exécutable ; ellipse≡fonction ; losange≡programme ; polygone≡fichier entrée/sortie ; étoile≡module de calibration

finalement choisie pour relier l'algorithme BC-DFO au modèle HyMAP. Elle a nécessité l'introduction d'un nouveau mode de lancement de HyMAP qui écrit la valeur de la fonction objectif dans un fichier, ce qui n'était pas fait jusqu'à présent.

3.5 Comparaison des deux approches

Les deux approches permettant d'intégrer la méthode d'optimisation BC-DFO codée en `Matlab` au sein du programme HyMAP écrit en `Fortran90` sont radicalement différentes. La première utilise un type de fichier appelé `mex-file`, codé en `C`, qui permet d'appeler la fonction `call_obj_func_bcdfo()` depuis la fonction `objf.m`. L'inconvénient est que l'appel du `mex-file` constitue un nouveau processus pour HyMAP, et cela requiert la déclaration de toutes les fonctions et variables dont est dépendante `call_obj_func_bcdfo()`. Pour que cela puisse fonctionner aisément, il faudrait que le code de HyMAP soit clairement structuré en composant IRF (Initialize/Run/Finalize). En effet, dans la version actuelle du code du modèle, les initialisations sont réparties dans de nombreux modules, ce qui rend difficile l'utilisation du `mex-file`. La seconde est plus simple car elle permet d'échanger les données entre les codes `Matlab` et les codes `Fortran90` par écriture/lecture dans des fichiers d'entrée/sortie. L'inconvénient de cette approche est justement l'utilisation de plus de mémoire pour stocker les fichiers. De plus, l'écriture et la lecture de fichiers requièrent un temps supplémentaire d'exécution. L'idéal est toutefois de se diriger vers un couplage de codes de type PALM.

L'approche choisie pour le reste du projet est la seconde car elle plus légère et plus adaptée au temps imparti à ce projet de modélisation. Cependant, il est à envisager de structurer le code du modèle HyMAP en IRF de sorte à faciliter le couplage de codes de types PALM et `mex-file`.

Partie 4

Résultats

4.1 Introduction

Les parties précédentes ont permis de présenter les aspects théoriques relatifs au projet de modélisation et de décrire la méthode que nous avons employée pour résoudre le problème posé : comment coupler l'algorithme d'optimisation BC-DFO codé en `Matlab` avec le modèle HyMAP dont les codes sources sont écrits en `Fortran90` ?

Dans cette partie, les résultats sont exposés pour montrer l'impact de la méthode mono-objectif locale sur la minimisation de la fonction objectif et sur le jeu de paramètre optimal. Une analyse de ces résultats est proposée sur la base de comparaisons des deux méthodes d'optimisation. Enfin, les résultats font l'objet d'une synthèse dans une conclusion.

4.2 Résultats de la calibration automatique

4.2.1 La calibration automatique par MOCOM-UA

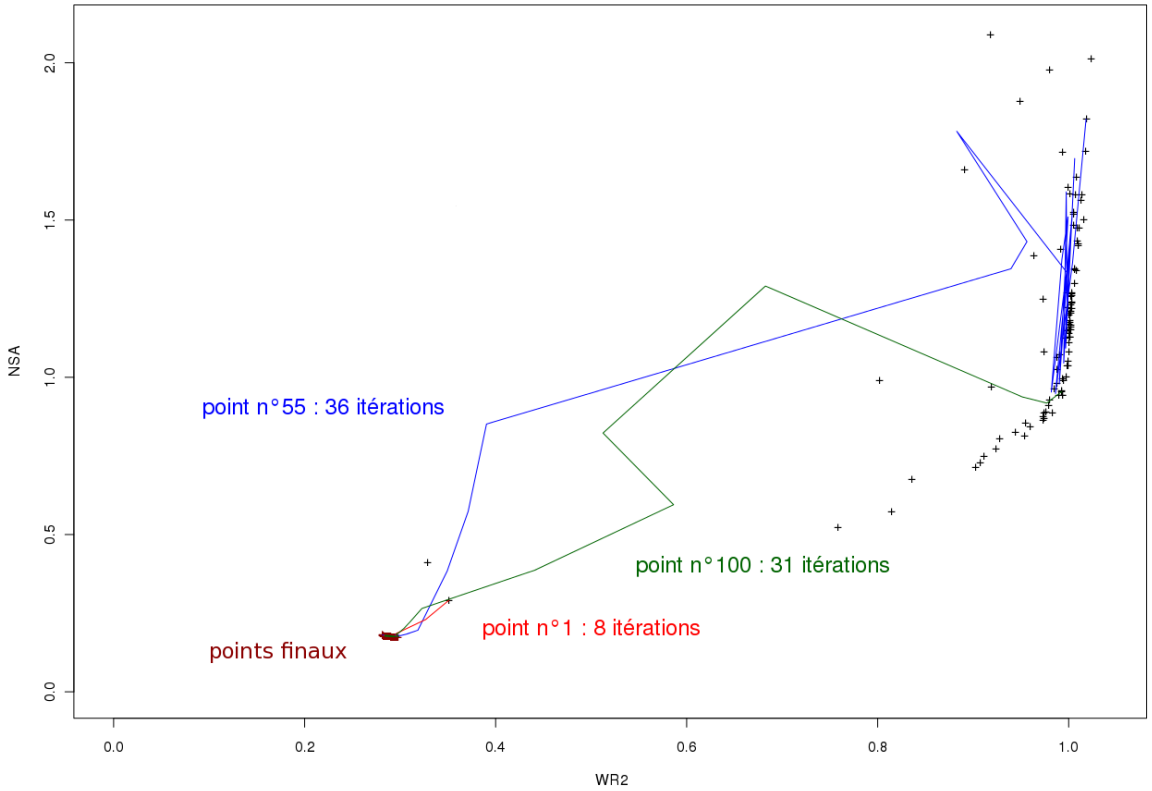
Conditions de l'essai

Conformément au principe de fonctionnement de l'algorithme d'optimisation MOCOM-UA, la calibration a été lancée avec cent points de départ correspondant à cent jeux de paramètres. Parmi cet ensemble de points, le premier est fixé à $(1, 1, 1, 1)$, cela correspondant à des valeurs qu'on estime représentatives de la réalité physique pour le bassin de l'Amazone. Les quatre-vingt dix-neuf autres jeux de paramètres sont échantillonnés aléatoirement dans l'espace des paramètres.

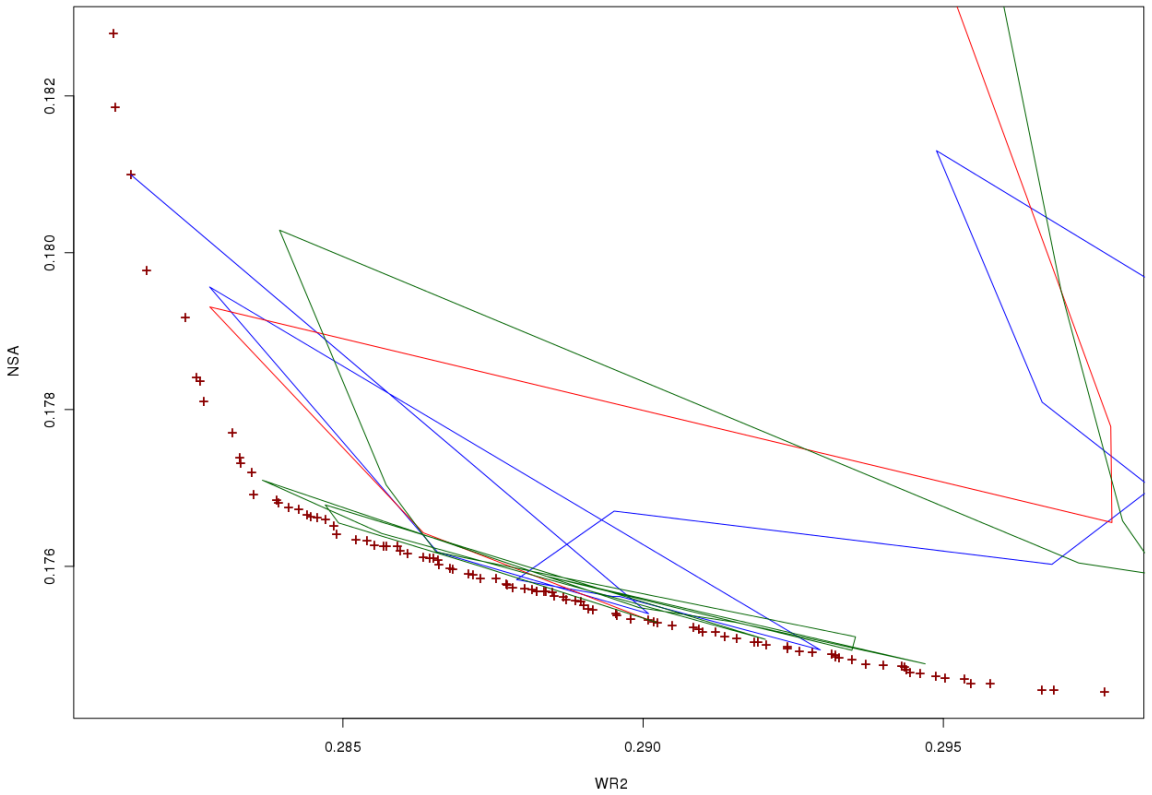
Résultats

La minimisation des deux fonctions objectif avec MOCOM-UA a requis 4501 évaluations de fonctions. À raison de 3 minutes par évaluation, l'optimisation a duré plus d'une semaine avant de converger.

Les résultats obtenus sont présentés sur la figure 4.1. On voit que les points de départ, quoique très dispersés, donnent finalement lieu à un ensemble ordonné : le front de Pareto (figure 4.1a). On remarque à l'aide de la figure 4.1a que le front de Pareto est concentré dans une zone étroite autour du minimum global. De plus, le front de Pareto est à la fois convexe et connexe comme attendu théoriquement (Fig. 4.1b). Ce résultat est important pour justifier l'utilisation de l'optimiseur BC-DFO qui utilise la somme des deux fonctions objectif. D'autre part, le fait que le front de Pareto soit concentré autorise l'emploi d'une méthode d'optimisation mono-objectif.



(a) Trajectoires des points dans l'espace des fonctions objectif. En noir : points d'origine ; en rouge foncé : points finaux ; en bleu, vert et rouge : trajectoires de 3 des points au cours de l'optimisation



(b) Zoom de la figure 4.1a mettant en valeur le front de Pareto final

FIGURE 4.1 – Trajectoires des points dans l'espace des fonctions objectif au cours des itérations de l'optimisation

Sur les figures 4.1a et 4.1b sont également représentées les trajectoires de quelques points, c'est-à-dire l'évolution de la position de ces points au cours des évaluations dans l'espace des fonctions objectif. Il est à constater que tous les points n'ont pas des trajectoires semblables. La courbe rouge vif correspond à la trajectoire du premier point, dont les valeurs des coefficients de départ sont tous à 1. On voit que ce point est initialement plus proche du front de Pareto que les autres points et que le nombre d'itérations qui lui ont été nécessaire pour rejoindre ce front est aussi peu élevé. La proximité du premier point au front de Pareto démontre la bonne représentation de la physique du bassin de l'Amazone pour HyMAP.

D'autre part, MOCOM-UA peut donner lieu à de nombreuses oscillations, d'où une certaine perte d'efficacité. En effet, si le point 100 parvient en début d'algorithme à se diriger rapidement vers le minimum à l'aide de réflections au sein de simplexes (comme expliqué dans la partie 2), il effectue des allers-retours au cours de nombreuses itérations suivantes (Fig. 4.1b, trajectoire verte). Au contraire, le point 55 stagne autour de sa position initiale (Fig. 4.1a, trajectoire bleue) sur de nombreuses itérations, avant de converger plus rapidement vers le minimum à partir du moment où le simplexe contient des points plus proches du minimum.

4.2.2 La calibration automatique par BC-DFO

Conditions des essais

Afin de tester la capacité de l'algorithme BC-DFO à converger vers un minimum proche du minimum global, nous avons effectué plusieurs tests avec différentes conditions initiales pour le vecteur des coefficients multiplicateurs des paramètres x. Pour le premier test, chaque coefficient est fixé à la valeur 1, tout comme cela est fait dans l'optimisation MOCOM-UA pour le point initial. Pour le second test, les coefficients multiplicateurs associés respectivement aux paramètres T_b, η_r, W et H ont été initialisés à $(15, 4, 1, 1)$. Ces valeurs sont assez proches des contraintes de bornes supérieures fixées respectivement à $(20, 5, 1.2, 1.2)$.

Résultats du premier test

La minisation de la fonction objectif par l'algorithme BC-DFO a requis 105 évaluations et 67 itérations soit environ huit heures et demi sur une machine de calcul standard.

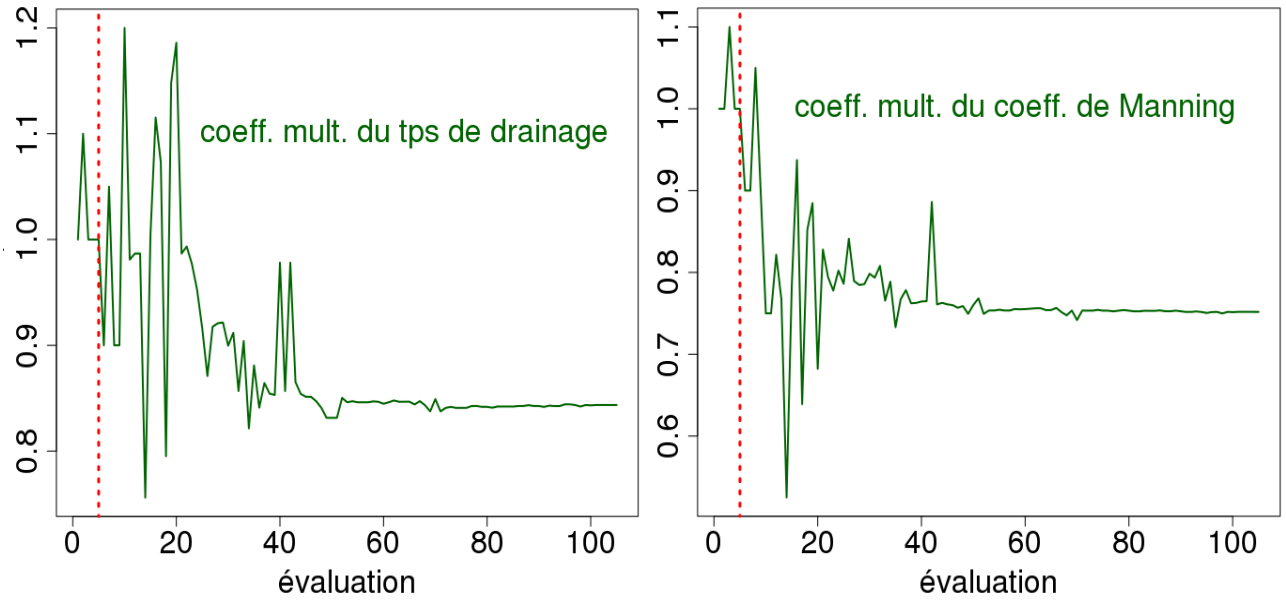
La figure 4.2 représente l'évolution de chacun des coefficients multiplicateurs des quatre paramètres (T_b, η_r, W et H, respectivement de la figure 4.2a à la figure 4.2d).

On observe que pour les trois coefficients multiplicateurs associés à T_b, η_r et H, chaque courbe oscille jusqu'à la cinquantième itération approximativement avant de se stabiliser autour de la valeur correspondant au jeu de paramètres optimal (respectivement 0.84, 0.75 et 0.74). La courbe représentant l'avancée du coefficient multiplicateur de W montre quant à elle une activation de la borne supérieure fixée à 1.2 dès la première itération.

Ce test est concluant car il montre bien la capacité de l'optimiseur à converger rapidement vers un minimum local proche du minimum global. En effet, il a requis 40 fois moins d'évaluations de fonction objectif que l'algorithme MOCOM-UA, ce qui représente un gain de temps de calcul considérable (tel qu'il sera vu dans la section 4.2.3).

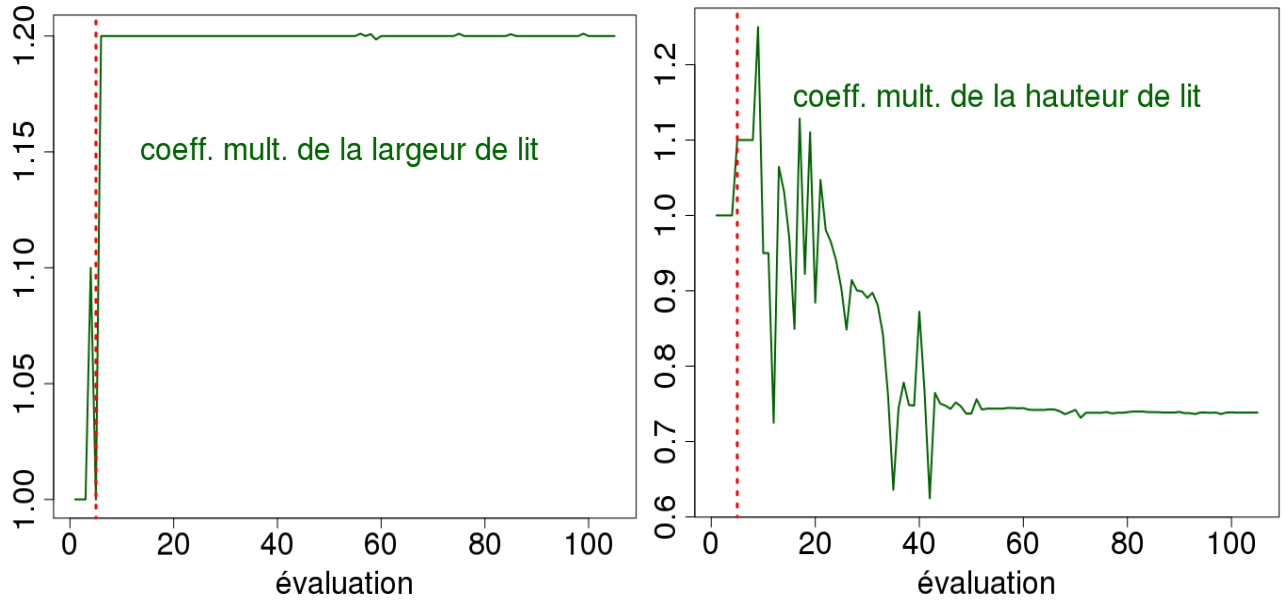
Résultat du second test

La figure 4.3 représente l'ensemble des valeurs successives de chacun des coefficients multiplicateurs des quatre paramètres pour lesquels la fonction objectif a été évaluée dans le second cas. Cet ensemble montre l'évolution des coefficients multiplicateurs. Il est à noter que certaines valeurs dépassent parfois les bornes. Ces valeurs ne sont pas reliées à des points itérés mais à des points apparus lors de nouveaux échantillonnages pour la construction de modèles lors du



(a) T_b

(b) η_r



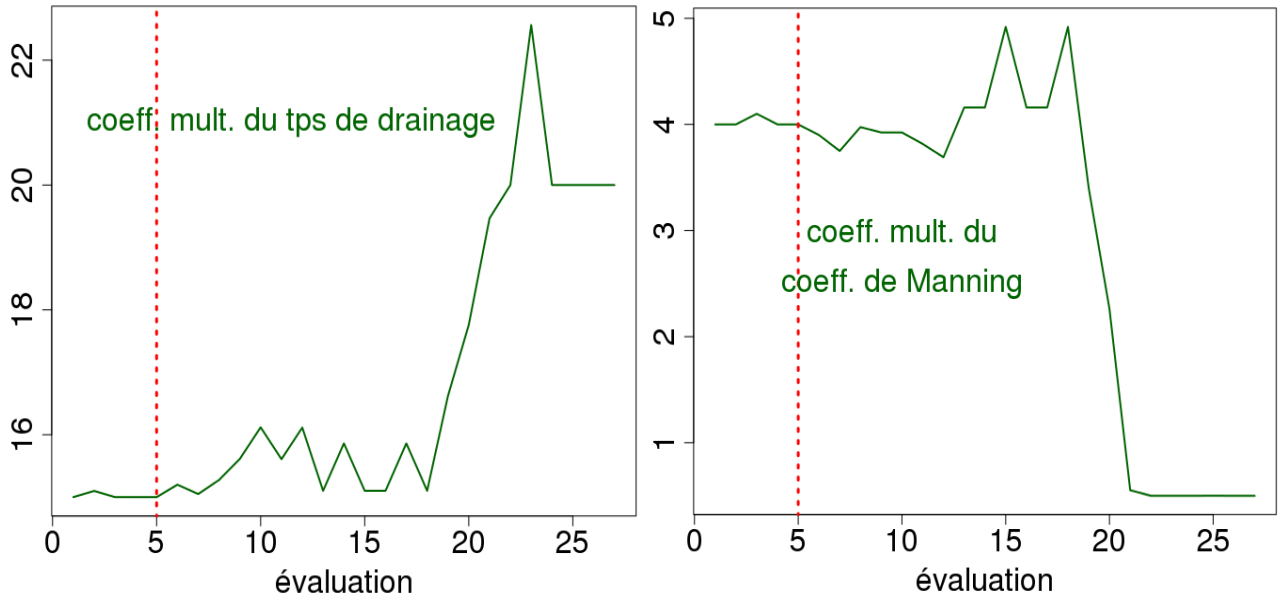
(c) W

(d) H

FIGURE 4.2 – Valeurs des quatre coefficients multiplicateurs associés aux paramètres en fonction de l'avancée des évaluations pour le premier test. La ligne rouge marque le début du processus d'optimisation après les cinq premières évaluations liées à l'échantillonnage initial.

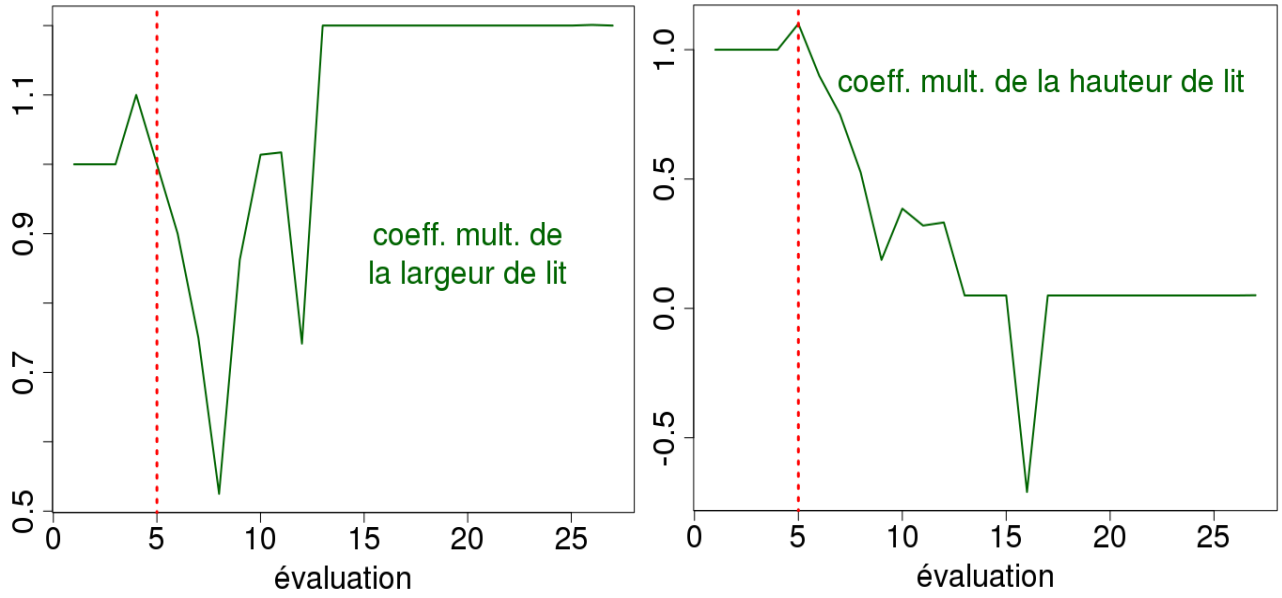
passage d'un sous-espace à l'autre (voir la section 2.3.2). Ces valeurs ne sont donc pas le reflet d'un défaut de l'algorithme.

On observe cette fois-ci que les quatre coefficients atteignent rapidement leur bornes. En effet, la minimisation a requis seulement 13 itérations et 27 évaluations de fonction objectif. Pour deux des coefficients, T_b et W, il s'agit des bornes supérieures (respectivement 20 et 1.2) alors que pour les deux autres, η_r et H, ce sont les bornes inférieures qui sont atteintes (respectivement 0.5 et 0.05). En voyant ces résultats, on peut penser que le minimum atteint est local ce qui sera confirmé dans la section 4.2.3, lors de la comparaison avec MOCOM-UA. De plus, cela a modifié considérablement la valeur de T_b et de H par rapport aux estimations des valeurs physiques qui sont données par le modèle de départ.



(a) T_b

(b) η_r



(c) W

(d) H

FIGURE 4.3 – Valeurs des quatre coefficients multiplicateurs associés aux paramètres en fonction de l'avancée des évaluations pour le second test. La ligne rouge marque le début du processus d'optimisation après les cinq premières évaluations liées à l'échantillonnage initial.

Ce test montre un aspect négatif de la minimisation mono-objectif appliquée au modèle HyMAP. En effet, le risque de voir BC-DFO converger vers un minimum local éloigné du minimum global existe réellement et doit être pris en compte.

4.2.3 Comparaison des deux calibrations automatiques

Pour comparer les résultats obtenus par les deux méthodes de calibration, il est intéressant de calculer les valeurs absolues des différences nettes et relatives des deux fonctions coût à l'optimum notées respectivement d et D par la suite et calculées comme suit :

$$d = |f_{MOCOM-UA} - f_{BC-DFO}|,$$
$$D = \frac{d}{f_{MOCOM-UA}}.$$

Il sera également intéressant de calculer les valeurs de décroissance relative des fonctions objectif entre BC-DFO et MOCOM-UA :

$$\text{décroissance relative} = \frac{|\delta f_{MOCOM-UA} - \delta f_{BC-DFO}|}{\delta f_{MOCOM-UA}},$$
avec δf la décroissance de la fonction objectif f : $\delta f = (f_{initiale} - f_{optimum})$.

Comparaison du premier test de la calibration BC-DFO avec MOCOM-UA

Le tableau 4.1 résume l'ensemble des résultats obtenus pour les deux types de calibration en partant d'un jeu de paramètres fixé à 1 (premier test).

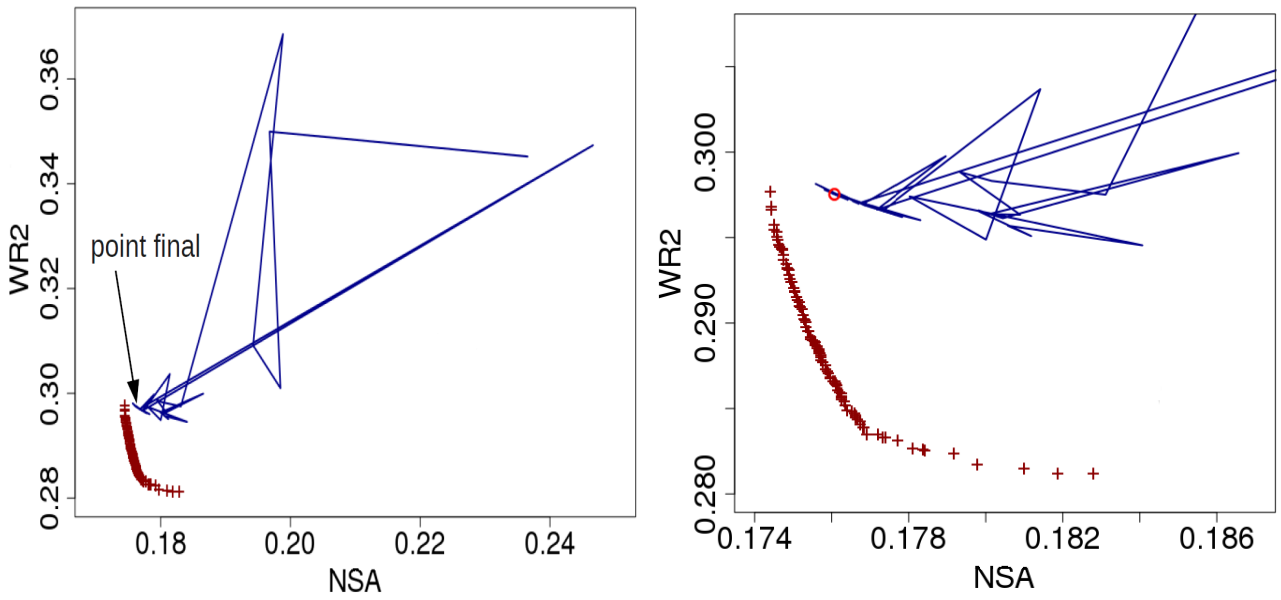
La convergence de la minimisation MOCOM-UA et le choix arbitraire du point milieu du front de Pareto donnent les valeurs respectives 0.2884 et 0.1757 pour WR^2 et NSA.

La convergence de la minimisation BC-DFO donne quant à elle les valeurs respectives 0.2975 et 0.1761 pour les deux mêmes fonctions coût. Pour la fonction coût NSA, $d_{NSA} = 0.0004$, et $D_{NSA} = 0.22\%$. Pour la fonction coût WR^2, $d_{WR^2} = 0.0091$, et $D_{WR^2} = 3.16\%$.

L'algorithme BC-DFO a nécessité 105 évaluations de fonction en 69 itérations pour converger, alors que la charge en coûts de calculs de MOCOM-UA s'est avérée bien plus élevée : 4501 évaluations de fonction en 1114 itérations.

Les différences relatives étant très faibles et l'algorithme MOCOM-UA convergeant vers le minimum global, on peut se satisfaire du résultat obtenu par l'algorithme BC-DFO. En effet, il converge vers un minimum local très proche du minimum global trouvé par MOCOM-UA. Ceci est illustré par la figure 4.4. Il est à remarquer que les valeurs des fonctions objectif à l'optimum sont proches, ce qui n'est pas nécessairement le cas des valeurs des coefficients multiplicateurs des paramètres (notamment T_b et H). Cependant, le but n'est pas ici de trouver un jeu physique de paramètres, ce n'est donc pas un réel problème.

Le calcul de la décroissance relative entre les algorithmes BC-DFO et MOCOM-UA accentue ce fait. En effet, elle vaut seulement 0.3% pour le critère NSA et 21% pour le critère WR^2 (la décroissance relative serait égale à 0% si BC-DFO convergeait totalement vers le minimum global). Il est à noter qu'il pourrait être souhaitable d'améliorer cette valeur pour le second critère lors de futurs tests.



(a) (b) Zoom de la figure 4.4a

FIGURE 4.4 – Trajectoire du point de minimisation BC-DFO dans l'espace des fonctions coût (en bleu) et front de Pareto (en rouge).

	MOCOM-UA	BC-DFO
Valeurs initiales des paramètres	$(1, 1, 1, 1)$	$(1, 1, 1, 1)$
Valeurs des paramètres à l'optimum	$(0.62, 0.70, 1.19, 0.43)$	$(0.84, 0.75, 1.2, 0.74)$
Valeurs initiales des fonctions objectif		
- WR^2	0.3514	0.3514
- NSA	0.2918	0.2918
Valeurs des fonctions objectif à l'optimum		
- WR^2	0.2884	0.2975
- NSA	0.1757	0.1761
Nombre total d'itérations	1114	69
Nombre total d'évaluations des fonctions objectif	4501	105
Décroissance des fonctions objectif		
- δWR^2	0.063	0.0539
- δNSA	0.1161	0.1157

Tableau 4.1 – Tableau récapitulatif des résultats du premier test de calibration par BC-DFO et de la calibration MOCOM-UA.

Comparaison du second test de la calibration BC-DFO avec MOCOM-UA

Le tableau 4.2 résume l'ensemble des résultats obtenus pour les deux types de calibration (second test). Le second test a été encore plus prompt à converger que le premier : seulement 27 évaluations de fonctione en 13 itérations.

La convergence de la minimisation BC-DFO donne les valeurs respectives 1.0600 et 1.0004 pour les fonctions coût NSA et WR^2. Pour la fonction coût NSA, $d_{NSA} = 0.8247$, et $D_{NSA} = 469.4\%$. Pour la fonction coût WR^2, $d_{WR^2} = 0.7716$, et $D_{WR^2} = 267.55\%$. Les différences relatives sont cette fois-ci très élevées. Cela dévoile la fragilité de l'algorithme BC-DFO. Les décroissances relatives pour ce test sont égales à 73.56% pour le critère NSA et à 29.28% pour le critère WR^2. Ces valeurs sont grandes et traduisent le fait que le minimum trouvé par BC-DFO

est loin du minimum trouvé par MOCOM-UA. Cependant, la valeur de la décroissance relative du second critère est assez proche du premier test. L'algorithme semble donc se comporter identiquement pour les deux tests vis-à-vis de ce critère. Cela accentue l'idée de pondérer la somme des fonctions objectif (comme vu dans la partie 2). Ce deuxième test montre aussi que l'algorithme est sensible aux positions initiales des points, ce qui peut l'entraîner à converger vers un minimum local très éloigné du minimum global (Fig. 4.5).

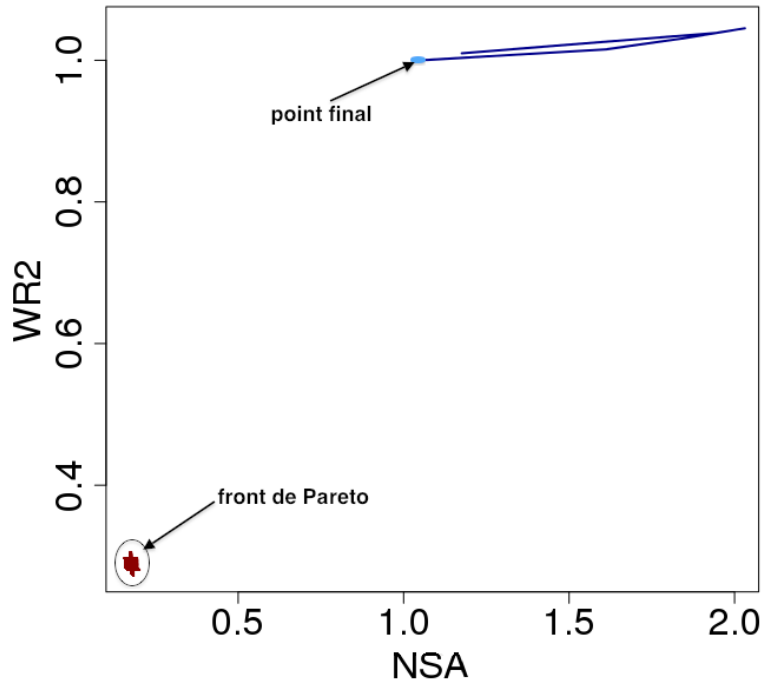


FIGURE 4.5 – Trajectoire du point de minimisation BC-DFO dans l'espace des fonctions coût (en bleu) et front de Pareto trouvé par MOCOM-UA (en rouge).

	MOCOM-UA	BC-DFO
Valeurs initiales des paramètres	$(1, 1, 1, 1)$	$(15, 4, 1, 1)$
Valeurs des paramètres à l'optimum	$(0.62, 0.70, 1.19, 0.43)$	$(20, 0.5, 1.2, 0.05)$
Valeurs initiales des fonctions objectif		
- WR^2	0.3514	2.3528
- NSA	0.2918	1.0307
Valeurs des fonctions objectif à l'optimum		
- WR^2	0.2884	1.0600
- NSA	0.1757	1.0004
Nombre total d'itérations	1114	13
Nombre total d'évaluations des fonctions objectif	4501	27
Décroissance des fonctions objectif		
- δWR^2	0.0630	1.2928
- δNSA	0.1161	0.0303

Tableau 4.2 – Tableau récapitulatif des résultats du second test de calibration par BC-DFO et de la calibration MOCOM-UA.

4.3 Conclusion

Par le biais du premier test effectué avec l'algorithme BC-DFO, les résultats nous montrent la capacité de cet algorithme à converger vers un minimum proche du minimum global et ceci en faisant d'importantes économies en temps de calcul (il converge en 105 évaluations de fonctions contre 4501 pour MOCOM-UA). Cependant, nous avons également constaté sa fragilité concernant la détection du minimum global. Il faut donc rester vigilant aux points initiaux fournis à l'algorithme.

Partie 5

Vers l'assimilation de données par BC-DFO

5.1 Introduction

Après avoir allégé le temps de calibration à l'aide de l'optimiseur BC-DFO, il est apparu intéressant d'utiliser ce même optimiseur pour l'assimilation de données. L'assimilation de données consiste à minimiser une fonction coût qui tient compte des erreurs de mesures liées aux instruments d'observations (ENVISAT) et des erreurs d'ébauches liées aux paramètres du modèle. Les statistiques de ces erreurs sont stockées dans la matrice de covariance des erreurs d'observations R et la matrice de covariance des erreurs d'ébauche B. Ici, nous cherchons à assimiler non pas les états du modèle mais les paramètres. Pour plus d'informations sur l'assimilation de données, le lecteur peut consulter l'ouvrage de G. Desroziers, 2007. Il est important de préciser que cette partie est une ouverture du projet de modélisation qui n'entrait pas initialement dans les objectifs. Le sujet de l'assimilation de données avec BC-DFO est donc un premier pas dans cette direction.

5.2 Définition de la fonction coût

La fonction coût $J(x)$, en assimilation de données, s'écrit comme la somme de deux termes. Le premier est proportionnel à la norme de l'écart entre le jeu de paramètres courant x et le jeux de paramètres de l'ébauche x_b (ici fixé à $(1, 1, 1, 1)$), pondérée par la matrice de covariance des erreurs d'ébauche B. Le second terme est quant à lui proportionnel à la norme de l'écart entre les observations y et le jeu de paramètres courant intégré au modèle puis projeté dans l'espace des observations à l'aide du modèle HyMAP H, pondérée par la matrice de covariance des erreurs de mesure R.

$$J(x) = (x - x_b)^t B^{-1}(x - x_b) + \sum_i (y^i - H(x))^t R^{-1}(y^i - H(x))$$

Dans notre cas, les matrices B et R sont diagonales, seules les variances sont considérées (pas de corrélation entre les erreurs). Cette approximation est forte mais nécessaire au premier stade de l'étude. Les variances de la matrice B sont fixées arbitrairement à 0.05 et 0.1 pour la matrice R.

5.3 Modifications apportées au code

La figure 5.1 représente les modifications apportées à la structure générale des codes.

Le calcul de la fonction coût se fait au sein d'une nouvelle sous-routine `Fortran90` : `assim_ bcdfo.f90`. Cette sous-routine est appelée par `call_obj_func_assim.f90` nouvellement écrite. Une variable de choix supplémentaire a dû être intégrée au sein de `mod_calib.f90` afin de pouvoir lancer le modèle en mode assimilation depuis le module `mocom.f90`. Le mode de communication entre l'optimiseur BC-DFO et le modèle HyMAP est le même que précédemment, l'écriture/lecture dans des fichiers d'entrée/sortie.

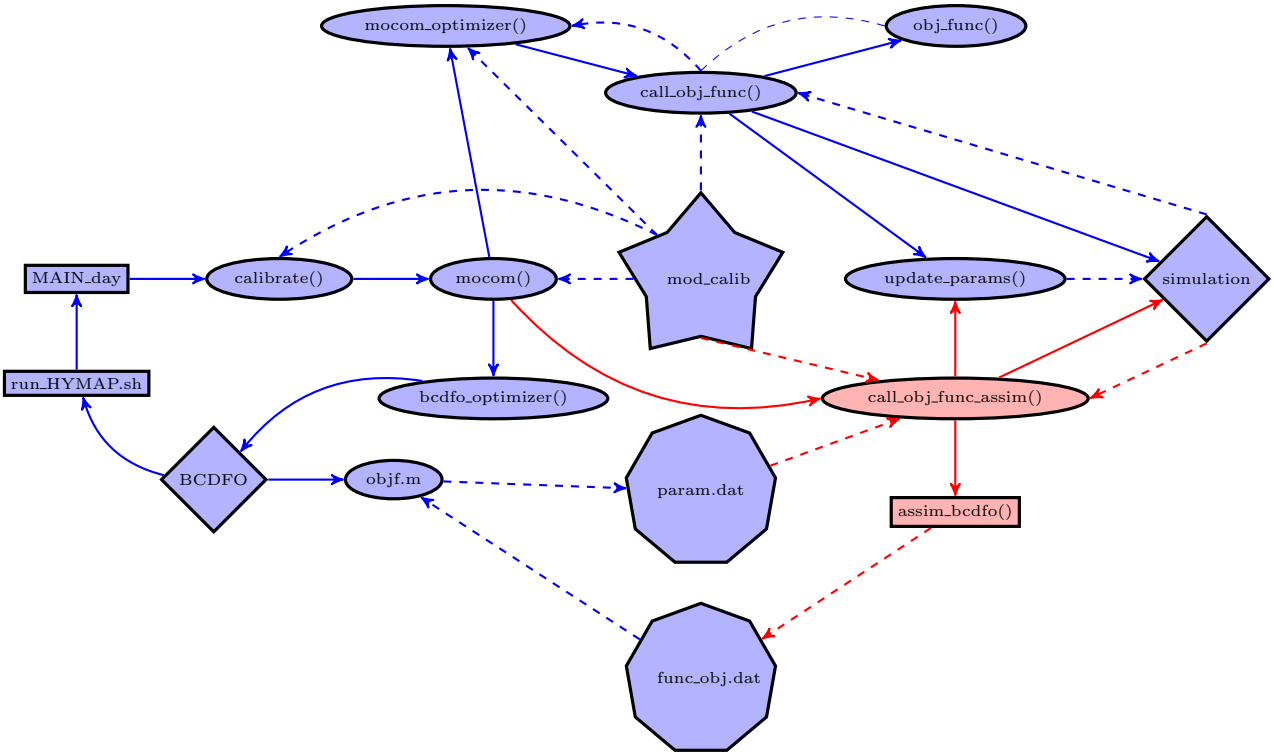


FIGURE 5.1 – Schéma de l'assimilation de données par BC-DFO. Flèche pleine≡appel d'une fonction ou d'un programme ; flèche tiretée≡transfert d'informations. Description des formes : rectangle≡exécutable ; ellipse≡fonction ; losange≡programme ; polygone≡fichier entrée/sortie ; étoile≡module de calibration

5.4 Résultats de l'assimilation de données appliquée aux paramètres de HyMAP

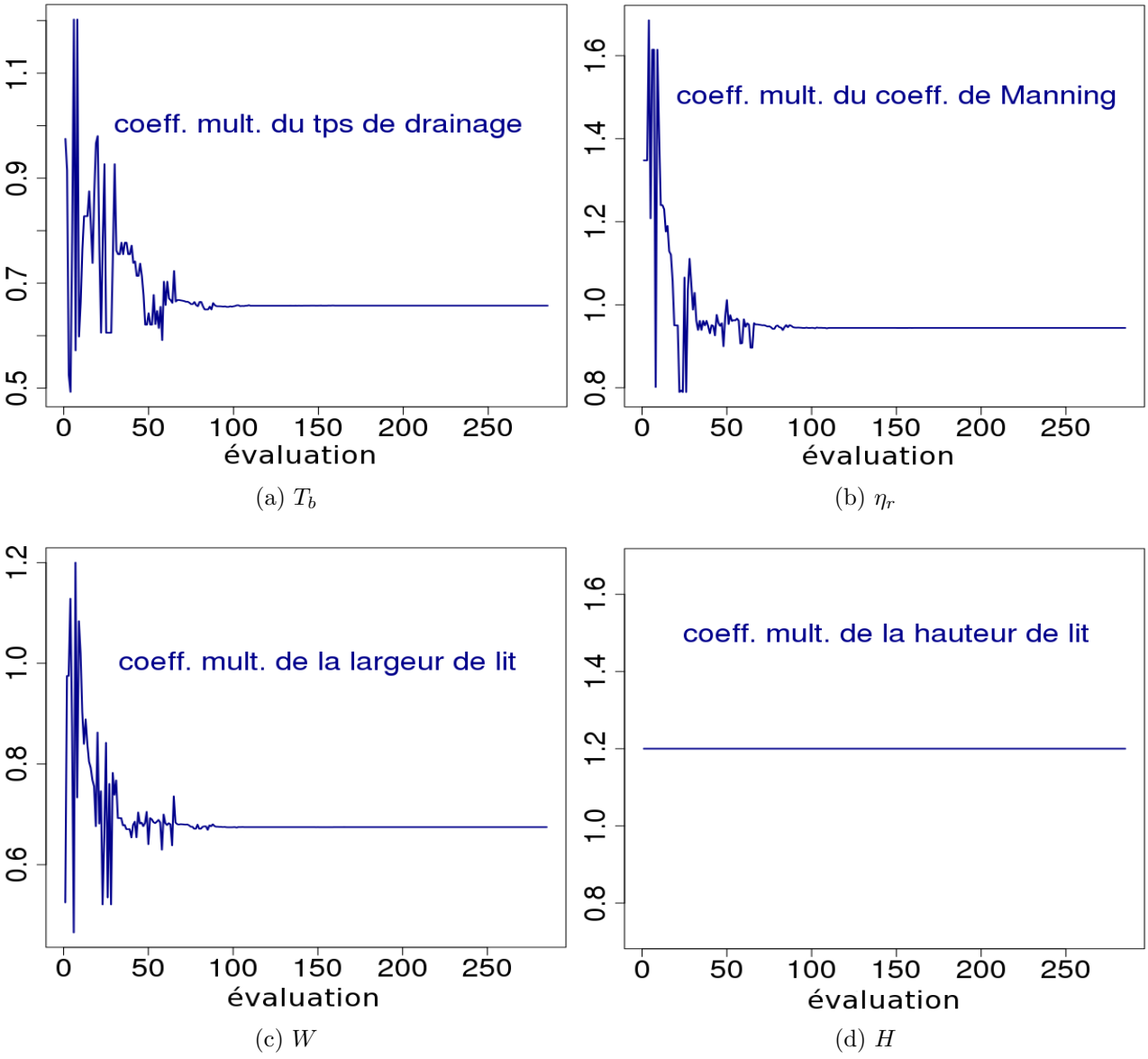
5.4.1 Conditions de l'essais

Un test a visé à lancer l'algorithme BC-DFO en mode assimilation de donnée avec un jeu de paramètres initial fixé à $(1, 1, 1, 1)$ et un nombre maximal d'itération initialisé à 300.

5.4.2 Résultats du test

La figure 5.2 représente l'avancée des quatre paramètres au cours des itérations de l'algorithme. Tout d'abord, le processus d'optimisation n'a pas convergé au bout des 300 itérations et a donc dû arrêter la minimisation. Cependant, les figures 5.2a à 5.2d montrent une stabilisation des valeurs des coefficients multiplcateurs des paramètres autour de $(0.6572, 0.9443, 0.6746, 1.2)$ pour T_b, η_r, W et H respectivement. On peut supposer que l'algorithme ait un problème de détection de la convergence pour la fonction objectif propre à l'assimilation de données. Ce

type de non-détection est connu de l'auteur de l'algorithme. La stabilisation des paramètres a eu lieu après 120 évaluations environ.



(a) T_b (b) η_r

(c) W (d) H

FIGURE 5.2 – Valeurs des quatres coefficients multiplicateurs associés aux paramètres en fonction de l'avancée des itérations.

5.5 Conclusion

Il est intéressant d'utiliser BC-DFO comme un outil d'assimilation de données. Ici, par manque de temps pour approfondir ce sujet, des résultats nous manquent pour interpréter physiquement les valeurs des paramètres optimaux et les comparer aux tests de la partie 4 avec les fonctions objectif. De plus, il faudra correctement modéliser les matrices de covariance des erreurs d'ébauche et d'observations. Des études plus approfondies sur l'adaptation de la fonction coût sont à réaliser.

Conclusion et bilan personnel

Conclusion sur le projet de modélisation

Le principal objectif de ce projet de modélisation, à savoir coupler une nouvelle méthode d'optimisation BC-DFO avec le modèle d'hygrologie globale HyMAP, a été atteint. Jusqu'à ce projet, les paramètres de ce modèle étaient calibrés par l'optimiseur multi-objectif évolutionnaire MOCOM-UA. Ce dernier permettait de converger vers un jeu de paramètres associés à un minimum global des deux fonctions coûts considérées (NSA et WR^2). Cependant, le fonctionnement propre à cet optimiseur impose un nombre considérable d'évaluations de ces fonctions coût, amenant le processus de calibration des paramètres à un coût de calcul élevé. C'est pour cela qu'il est apparu nécessaire de coupler le modèle HyMAP avec le nouvel optimiseur BC-DFO, moins coûteux car il converge plus rapidement, mais vers un minimum local. Ce couplage devait être fait de la manière la moins intrusive possible vis-à-vis du modèle HyMAP.

Différentes approches ont été considérées pour coupler l'optimiseur BC-DFO avec le modèle HyMAP, la principale difficulté étant qu'ils sont tous deux écrits dans des langages de programmation différents (`Matlab` pour l'optimiseur, `Fortran` pour le modèle d'hydrologie). Compte tenu du temps imparti au projet de modélisation, l'approche qui a été retenue pour mener à bien la communication entre le modèle et l'optimiseur est l'écriture/lecture de fichiers d'entrée/sortie échangés par les deux codes. À moyen terme, il serait préférable de s'orienter vers un couplage de code *via* PALM, car l'écriture/lecture de fichiers est un processus lourd en capacité mémoire et en temps de calcul.

Les résultats montrent plusieurs aspects intéressants de l'optimiseur nouvellement couplé au modèle. D'abord, cet optimiseur est capable de converger vers un minimum local proche du minimum global. En effet, les valeurs des erreurs relatives des fonctions objectif entre BC-DFO et MOCOM-UA sont assez faibles pour le premier test effectué sur BC-DFO (0.22% pour le critère NSA et 3.16% pour le critère WR^2). De plus, la convergence vers ce minimum est effectuée en un temps considérablement plus court que celui requis par l'optimiseur MOCOM-UA. Le nombre d'itérations est en effet de seulement 67 pour le premier test effectué sur BC-DFO contre 1114 pour MOCOM-UA. Cependant, nous avons également constaté la sensibilité de l'optimiseur BC-DFO aux conditions initiales. En effet, l'algorithme BC-DFO a convergé vers un minimum local éloigné du minimum global lors d'un second test effectué sur l'optimiseur, cela étant illustré par les fortes valeurs des erreurs relatives (469.4% pour le critère NSA et à 267.55% pour le critère WR^2). Il faudra donc être vigilant par la suite avec l'utilisation de cet optimiseur.

L'assimilation de données avec l'optimiseur BC-DFO est à approfondir avec intérêt à la suite de ce projet. En effet, cela pourrait permettre de converger vers un minimum en tenant compte des erreurs de mesures liées aux instruments et des erreurs d'ébauche liées au modèle.

Bilan personnel

Ce stage nous a permis de toucher du doigt de nombreuses facettes de la modélisation numérique.

Ainsi, nous avons tout d'abord entrevu le fonctionnement d'un modèle d'hydrologie de grande échelle, le modèle HyMAP. Ce modèle est encore en phase expérimentale, et notre travail s'est inscrit dans cette phase d'expérimentation et d'amélioration. Ce projet regroupe différents chercheurs, notamment en hydrologie et en optimisation.

C'est justement sur ce deuxième domaine que notre travail a principalement porté. En enffet, l'optimisation trouve de nombreuses applications dans les problématiques actuelles. C'est un sujet à la fois général et pointu, et c'est en partie ce qui nous a attiré vers ce projet. Du point de vue théorique, nous avons trouvé intéressant de découvrir deux méthodes différentes d'optimisation. Ces deux méthodes, qui ont chacune leurs forces et leurs faiblesses, sont relativement complexes et récentes.

Enfin, nous avons également pu appliquer un peu mieux la thématique de l'assimilation vue en cours à l'ENM. Ce champ de la modélisation est utile pour minimiser à la fois les erreurs d'ébauche et les erreurs d'observation, notamment en météorologie opérationnelle.

Pour finir, nous avons pu approfondir et mettre en pratique nos compétences en informatique, en modifiant et en debuggant des programmes relativement complexes.

Nous avons également tiré profit de la dimension de ce travail en équipe, où chacun a pu exprimer au mieux ses qualités et ses compétences.

Table des figures

Bibliographie

Desroziers, G. : *Implementation, diagnostic and optimisation of data assimilation schemes*, HDR, 2007.

Ehrgott, M. : *Multicriteria Optimization*, Lecture Notes in Economics and Mathematical Systems, Springer, Second ed, 2005.

Getirana, A.C.V., Boone, A., Yamazaki, D., Mognard, N. : *Automatic calibration of global flow routing scheme parameters driven by spatial altimetry data : evaluation in the Amazon basin.*

Getirana, A.C.V., Boone, A., Yamazaki, D., Decharme, B., Mognard, N. : *Hydrological Modeling and Analysis Platform (HyMAP)*, 2011.

Gratton, S., Toint, Ph.L., Troeltzch, A. : *An active-set trust-region method for derivative-free nonlinear bound-constrained optimization*, Optimization Methods and Software, vol. 21(4-5), pp. 873-894, 2011.

Logist, F., Vallerio, M., Van Impe, J. : *Explicit weight selection procedure for optimal control problems with weighted objectives*, Preprints of the 18$^{\text{th}}$ IFAC World Congress, 2011.

Nocedal, J., Wright, S. J. : *Numerical Optimization*, Series in Operations Research, SPRINGER, 1999.

Noilhan, J., Mahfouf, J.-F. : *The ISBA land surface parameterization scheme*, Global and Planetary Change, 13, 15p., 1996.

Yamazaki, D., Kanae, S., Kim, H., Oki, T. : *A physically based description of floodplain inundation dynamics in a global river routing model*, Water Resources Research, 21p., 2011.

Yapo, P.O., Gupta, H.V., Sorooshian, S. : *Multi-objective global optimization for hydrologic models*, Journal of Hydrology, 15p., 1998.