UQAC

Université du Québec
à Chicoutimi

8INF914 – Résolution de problèmes industriels

Projet de session

Partie II : Programmation dynamique

Auteurs :
Arthur Lebaron
Maxime Chevalier

Encadrant :
Pr. Sara Séguin

Version 1.0 du
28 mars 2018

Table des matières

Introduction

Rio Tinto est l'entreprise minière et métallurgique la plus importante au Canada. Près de 15000 employés sont répartis sur les 35 sites du groupe réparti dans le pays. Cependant, la métallurgie (production d'aluminium et de fer notamment) est une activité très énergivore. Pour pallier à des dépenses énergétiques trop importantes, les compagnies privées et le gouvernement ont très tôt investi dans la production hydroélectrique, notamment au $20^{\text{ème}}$ siècle. Aujourd'hui, cette production représente 97% de l'électricité produite au Québec.

Avec le rachat d'Alcan, le groupe Rio Tinto a acquis différentes centrales hydroélectrique qu'il utilise pour alimenter ses centrales. Cependant, ce type de ressource nécessite une gestion particulière car tout dépend de l'eau disponible. Les ingénieurs font face aux aléas de la nature. En effet, ils sont dépendants des stocks d'eau et des précipitations pour gérer efficacement la production énergétique et donc le fonctionnement de leurs métallurgies. En hiver, les précipitations sont sous forme solide (c'est une réserve non disponible sur le moment) qui permet de remplir les réservoirs lors de la fonte des neiges. Le reste de l'année, ils sont dépendants de leurs réserves et de la pluie. Pour maximiser leurs réserves, le groupe à continuer d'investir afin de contrôler toujours plus de vallées et ainsi récupérer l'écoulement d'eau.

Afin de construire leurs modèles de gestion de la production hydroélectrique, les ingénieurs disposent d'énormément de données. Un réseau de capteurs pluviométriques, ainsi que des capteurs de débits dans les rivières sont installés, leur permettant de prévoir l'arrivée d'eau dans les réservoirs et barrages, de même que la température de l'air (acquise par satellite) et toutes les données des années précédentes. Grâce à cela ils ont pu construire un modèle permettant de gérer efficacement la production hydroélectrique.

Pascal Cote, ingénieur analyste en recherche opérationnelle chez Rio Tinto Alcan, nous a fourni dans le cadre de ce projet les données de la centrale des Chutes du Diables (construit en 1953) datant de 2013. Nous avons donc à disposition, et ce pour chaque heure les données suivantes :

En-tête fichier de données	**Description**
Elav	Élévation aval
Qtot	Débit total à la centrale
Qvan	Débit déversé à la centrale
Niv Amont	Élévation amont
Q1, Q2, Q3, Q4, Q5	Débit turbiné par chaque turbine
P1, P2, P3, P4, P5	Puissance produite par chaque turbine

La centrale des Chutes du Diable est composée de 5 turbines, ayant chacune une capacité de production différente. Dans le rapport précédent, nous avons approximé la fonction de production associée à chaque turbine de la centrale afin de créer un modèle. Ce dernier nous permettra dans la suite de ce rapport de résoudre un système par programmation dynamique, qui prend en entrée le débit total à turbiner à la centrale ainsi que l'élévation amont, et qui retourne un débit pour chaque turbine et la production d'électricité associée. Ce système et son implémentation vont être décrit dans la suite de ce rapport.

Explication de la programmation dynamique

La programmation dynamique est une méthode générale de résolution de problème qui fonctionne de manière récursive. Cette méthode a été introduite en 1950 par Bellman et s'applique à des problèmes ayant des propriétés structurelles. Cette méthode s'appuie sur le principe de sous-optimalité défini par Bellman. Le problème de base P est décomposé en sous-problèmes $P_1, ..., P_k$, puis la solution optimale de chaque sous-problème est calculée. L'optimum sur P s'obtient ensuite à partir des optimaux des sous-problèmes. Ainsi, l'optimum pour une instance I peut se construire à partir de solutions optimales sur des instances plus simple $I_1, ..., I_k$

$$OPT(I) = f(OPT(I_1), ..., OPT(I_k))$$

On a donc une formulation récursive de l'optimum $OPT(I)$. Il suffit de calculer l'optimum pour $OPT(I_1), ..., OPT(I_k)$ puis d'appliquer f. Chaque $OPT(I_j)$ s'exprime à son tour en fonction d'instances plus simples jusqu'à obtenir une instance de base $\underline{I}$ directement calculable.

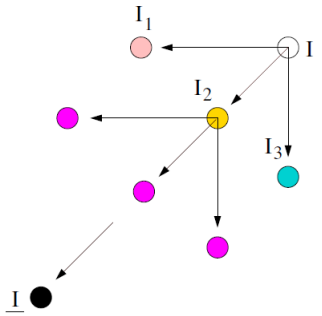

FIGURE 1.1 – Calcul récursif de l'optimum

En programmation dynamique, on appelle les sous-problèmes des noeuds et la dépendance entre ces sous problèmes des arêtes. C'est une structure DAG (Directed Acyclic Graph) qui est sous-jacente à tous les problèmes de programmation dynamique. Dans un DAG tous les noeuds peuvent être mis sur une ligne et tous les arcs vont de gauche à droite.

En se basant sur cette représentation en graph, on appelle :
— Une étape est un sous problème
— Chaque étape comporte des états

— Une décision est prise à chaque étape
— La décision affecte l'état de la prochaine étape

Il y a deux phases principales en DP (Dynamique Programing). La phase arrière et la phase avant.

— Arrière : On part du problème $\underline{I}$ et on résout chaque étape dans le sens inverse jusqu'à I_1. Cette étape permet de construire les tables de la programmation dynamique, qui contiennent toute l'information nécessaire à la construction de la solution optimale.
— Avant : On avance d'une étape à la fois pour construire la solution optimale*, en utilisant les tables définies à l'étape précédente. On part du premier sous-problème, en prenant la solution optimale construite et en soustrayant la solution optimale du sous-problème suivant et ainsi de suite. On obtient ainsi la solution de chaque sous-problème, et donc la solution au problème.

La programmation dynamique repose sur un paradigme très efficace. En effet, cette méthode ne nécessite pas de conditions sur la forme de la fonction objectif. Seule la propriété de sous-optimalité doit être vérifiée. Cependant, c'est une technique récursive, elle est donc gourmande en mémoire et devient inopérante si l'espace des états est grand. Il faut donc au maximum dérécursiver au maximum les sous-problèmes, stocker seulement l'information nécessaire pour calculer l'optimum et utiliser au plus les propriétés de dominances pour diminuer le nombre d'états.

On utilise cette méthode pour résoudre les problèmes de type :

— Plus courts chemins
— Problème de stockage
— Répartition de ressources
— Problèmes de remplacement d'équipement
— ...

Méthodologie retenue dans le cadre du projet

Dans le devoir précédent, nous avons obtenu les représentations analytiques des fonction de puissance, et de l'élévation aval en fonction de l'élévation amont et du débit total turbiné par la centrale. Grâce à ces représentations, nous devons modéliser et implémenter un algorithme de programmation dynamique, qui, selon le débit total à répartir, doit déterminer les turbines à mettre en marche ainsi que leur débit. Nous devons aussi prendre en compte certaines limitations, comme l'arrêt d'une turbine (pour une maintenance par exemple) ou une limitation du débit maximum possible pour celle-ci.

2.1 Modélisation du problème

Dans cette partie, nous visons à maximiser la puissance produite en fonction du débit à répartir afin d'aller chercher l'efficacité des turbines.

Données :
— Q_{tot} le débit total à répartir
— e_{amont} l'élévation amont au moment de la simulation
Variables :
— Q_i Débit à allouer à la turbine i
— e_{aval} Élévation aval calculé grâce à la modélisation faite au devoir 1
— h_{chute} Hauteur de chute de la turbine calculé grâce à la modélisation du devoir 1
Étape : Chaque turbine : $n \in \{1, ..., 5\}$ État : S_n Débit restant à attribuer à l'étape n

$$Max \sum_{i=1}^{5} P_i \quad [1]$$

s.c :

$$0 \leq \sum_{i=1}^{5} Q_i \leq Q_{tot}$$

$$\left.\begin{array}{l} e_{aval\,min} \leq e_{aval} \leq e_{aval\,max} \\ e_{amont\,min} \leq e_{amont} \leq e_{amont\,max} \end{array}\right\} = [2]$$

$$Q_{i\,min} \leq Q_i \leq Q_{i\,max} \forall i \in \{1, ..., 5\}$$

[1] avec $P_i = ((e_{am,i} - e_{av,i}(Q_{tot})) - 0.5e^{-5}\, Q_i^2) \times \mu(Q_i) \times Q_i$

[2] Ces contraintes sont sous-entendues car l'opérateur entre le débit à répartir

D'une étape à l'autre, nous avons la formule de récursion suivante :

$$f_5^*(S_5) = max\, P_5(Q_5)$$

$$\begin{aligned} f_4^*(S_4, Q_4) &= max\, P_4(Q_4) + f_5^*(S_4 - Q_4) \\ f_3^*(S_3, Q_3) &= max\, P_3(Q_3) + f_4^*(S_3 - Q_3) \\ f_2^*(S_2, Q_2) &= max\, P_2(Q_2) + f_2^*(S_2 - Q_2) \\ f_1^*(S_1, Q_1) &= max\, P_1(Q_1) + f_5^*(S_1 - Q_1) \end{aligned}$$

Contraintes supplémentaires :

— Nous devons prendre en compte le fait qu'une turbine soit en maintenance, et donc qu'elle ne soit pas disponible, ou disponible avec un débit limité.
— Si le débit est plus important que le débit maximal turbinable possible, nous devons déverser le reste.
— Discrétisassions des données : $5\,m^3/s$
— Débit minimal pour une turbine $0\,m^3/s$
— Débit maximal pour une turbine $180\,m^3/s$

2.2 Choix du langage de programmation

Nous avons choisi d'utiliser le langage de programmation JAVA, car c'est le langage objet où nous étions le plus à l'aise. De plus, l'application pourra être utilisées sur toutes les machines disposant d'une JVM.

Résultats

Dans cette partie, nous allons étudier les différents résultats obtenus lors des simulations. Pour effectuer les tests, nous avons choisi d'utiliser les lignes 1500 à 1700 du fichier Excel contenant les données de la centrale des Chutes du Diable. En effet, dans cet intervalle de ligne, chaque turbine est arrêté au moins une foi, et toutes les turbines sont activées en même temps au moins une foi. C'est donc pour nous un bon échantillon de test.

Nous avons procédé à deux évaluations. En effet, les données de base ne spécifient pas, lorsqu'une turbine est arrêtée, la raison. Nous ne savons donc pas, si c'est dû au modèle (pour maximiser le débit) ou si c'est pour cause de maintenance. Nous avons donc considéré les deux cas, afin de voir l'impacte que cela produit sur notre simulation.

Afin de faciliter la production de résultats, notre programme lit les données dans un fichier Excel, et écrit les résultats sur une nouvelle feuille dans le même classeur. Ainsi, nous avons pu facilement créer des graphiques.

3.1 Turbine 1

3.1.1 Test sans maintenance

Nous avons obtenu le résultat suivant lorsque nous n'avons pas fait de supposition sur la valeur d'un 0 dans le fichier de données.

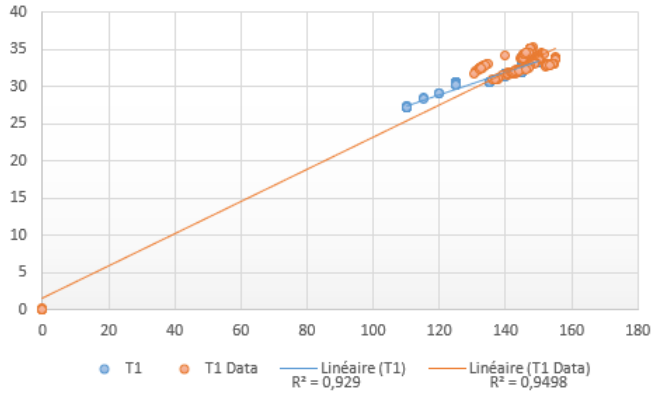

FIGURE 3.2 – Turbine 1 : Comparaison entre les résultats du modèle (en bleu) et les données réelles (en orange), avec le débit alloué en abscisse et la puissance générée en ordonnée

On remarque ici que la turbine 1 n'est jamais inutilisée. Son débit reste dans le bon ordre de grandeur.

3.1.2 Test avec maintenance

Nous avons obtenu le résultat suivant lorsque nous avons supposé qu'un 0 dans les données signifiait que la turbine était en maintenance.

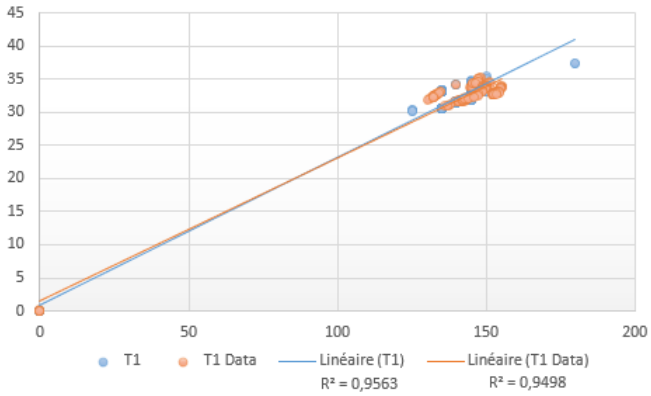

FIGURE 3.3 – Turbine 1 : Comparaison entre les résultats du modèle (en bleu) et les données réelles (en orange), avec le débit alloué en abscisse et la puissance générée en ordonnée

En considérant les 0 comme une maintenance, on voit que les données générées par le programme sont plus proches de l'exemple. Il n'y a qu'un seul out-lier qui est due à une borne supérieure trop importante pour la turbine.

3.2 Turbine 2

3.2.1 Test sans maintenance

Nous avons obtenu le résultat suivant lorsque nous n'avons pas fait de supposition sur la valeur d'un 0 dans le fichier de données.

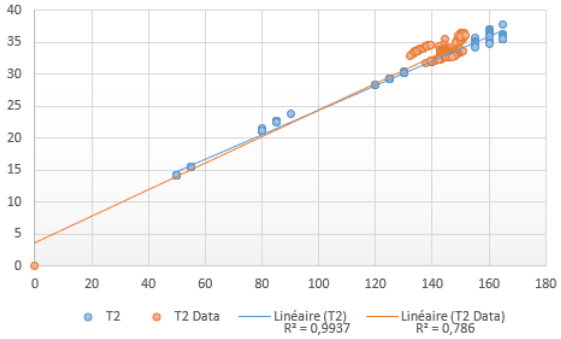

FIGURE 3.4 – Turbine 2 : Comparaison entre les résultats du modèle (en bleu) et les données réelles (en orange), avec le débit alloué en abscisse et la puissance générée en ordonnée

Ici non plus, la turbine n'est jamais mise à 0 par la simulation. La répartition est également étrange, en comparaison des données réelles. La turbine est utilisée à des débits où le rendement n'est pas optimal.

3.2.2 Test avec maintenance

Nous avons obtenu le résultat suivant lorsque nous avons supposé qu'un 0 dans les données signifiait que la turbine était en maintenance.

Même en considérant des opérations de maintenance, qui avait amélioré les résultats de la turbine 1, nous obtenons le même comportement. La turbine 2 à l'air d'être la moins puissante et de servir de buffer pour les autres turbines. Il est quand même étrange de ne pas fermer la turbine, par exemple quand elle est sous les $100\,m^3/s$ pour répartir l'eau sur les autres turbines et donc aller chercher le maximum d'efficacité.

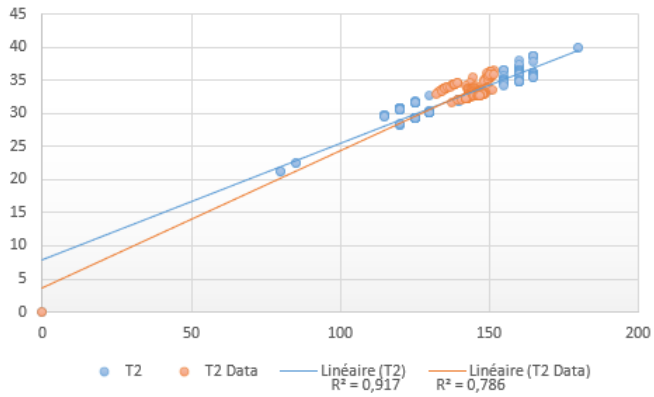

FIGURE 3.5 – Turbine 2 : Comparaison entre les résultats du modèle (en bleu) et les données réelles (en orange), avec le débit alloué en abscisse et la puissance générée en ordonnée

3.3 Turbine 3

3.3.1 Test sans maintenance

Nous avons obtenu le résultat suivant lorsque nous n'avons pas fait de supposition sur la valeur d'un 0 dans le fichier de données.

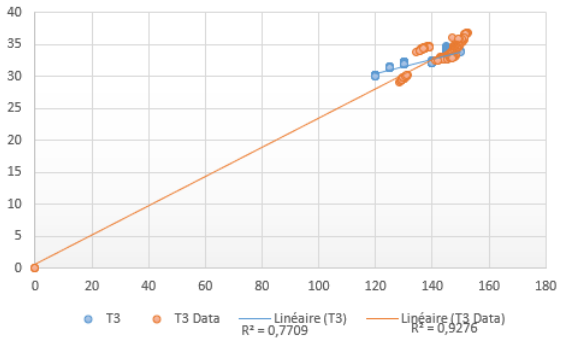

FIGURE 3.6 – Turbine 3 : Comparaison entre les résultats du modèle (en bleu) et les données réelles (en orange), avec le débit alloué en abscisse et la puissance générée en ordonnée

Encore une foi la turbine 3 n'est jamais mise à 0. Cependant la fonction à l'air de produire de bons résultats, car les points sont dans la même zone.

3.3.2 Test avec maintenance

Nous avons obtenu le résultat suivant lorsque nous avons supposé qu'un 0 dans les données signifiait que la turbine était en maintenance.

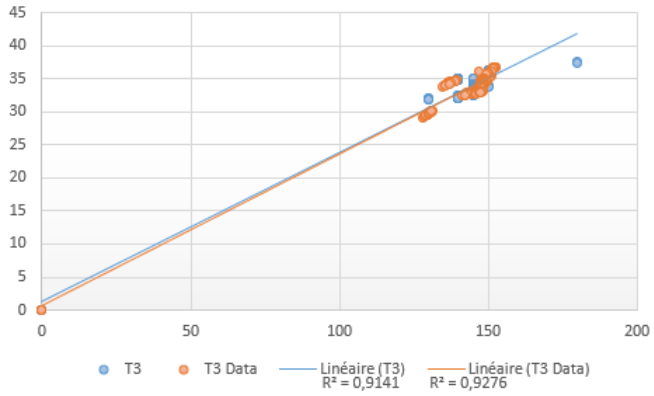

FIGURE 3.7 – Turbine 3 : Comparaison entre les résultats du modèle (en bleu) et les données réelles (en orange), avec le débit alloué en abscisse et la puissance générée en ordonnée

Comme pour la turbine 1, nous avons des résultats qui semblent en accord avec la fonction de production réelle. Il y a également un out-lier due à une borne supérieure trop importante.

3.4 Turbine 4

3.4.1 Test sans maintenance

Nous avons obtenu le résultat suivant lorsque nous n'avons pas fait de supposition sur la valeur d'un 0 dans le fichier de données.

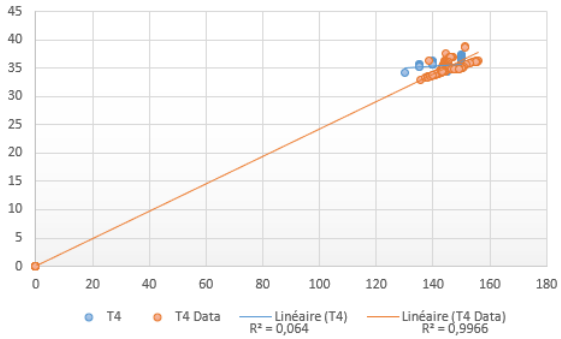

FIGURE 3.8 – Turbine 4 : Comparaison entre les résultats du modèle (en bleu) et les données réelles (en orange), avec le débit alloué en abscisse et la puissance générée en ordonnée

Idem que pour les turbines 1 et 3.

3.4.2 Test avec maintenance

Nous avons obtenu le résultat suivant lorsque nous avons supposé qu'un 0 dans les données signifiait que la turbine était en maintenance.

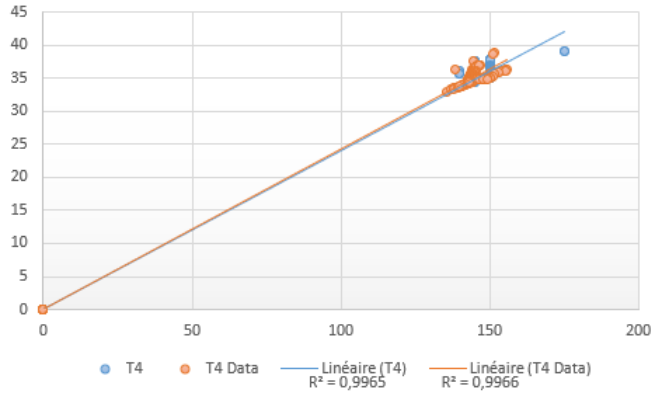

FIGURE 3.9 – Turbine 4 : Comparaison entre les résultats du modèle (en bleu) et les données réelles (en orange), avec le débit alloué en abscisse et la puissance générée en ordonnée

Idem que pour les turbines 1 et 3.

3.5 Turbine 5

3.5.1 Test sans maintenance

Nous avons obtenu le résultat suivant lorsque nous n'avons pas fait de supposition sur la valeur d'un 0 dans le fichier de données.

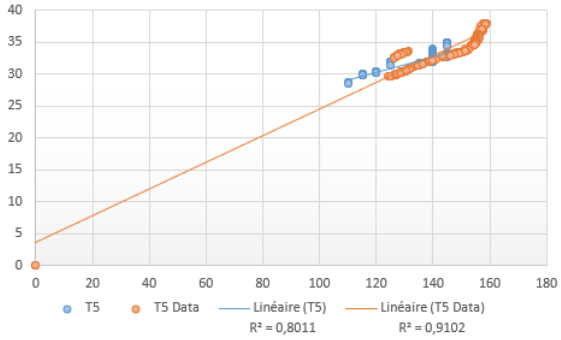

FIGURE 3.10 – Turbine 5 : Comparaison entre les résultats du modèle (en bleu) et les données réelles (en orange), avec le débit alloué en abscisse et la puissance générée en ordonnée

La turbine 5, dont nous n'avions pas réussi à trouver une modélisation fidèle, paraît sous-évaluée. En effet, notre modèle ne va pas chercher les valeurs d'efficacité optimale (autour de $160\,m^3/s$). De ce fait, il est possible que la répartition qui irait normalement à cette turbine aille vers une autre turbine (la deuxième par exemple qui à l'air d'être une turbine tampon).

3.5.2 Test avec maintenance

Nous avons obtenu le résultat suivant lorsque nous avons supposé qu'un 0 dans les données signifiait que la turbine était en maintenance.

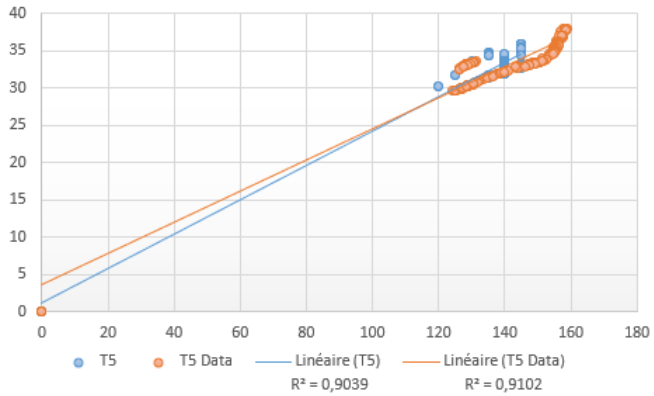

FIGURE 3.11 – Turbine 5 : Comparaison entre les résultats du modèle (en bleu) et les données réelles (en orange), avec le débit alloué en abscisse et la puissance générée en ordonnée

Avec la maintenance, les résultats ne changent pas énormément. C'est toujours le même souci d'optimum non trouvé.

3.6 Production totale

Nous nous intéressons maintenant à l'impacte de nos modélisation sur la production électrique totale. Toujours en comparant les données selon les deux méthodes, nous obtenons les résultats suivants :

3.6.1 Test sans maintenance

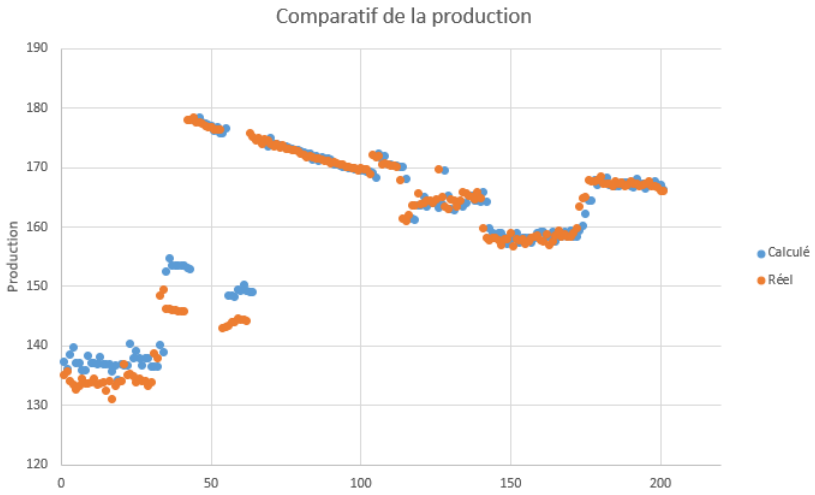

FIGURE 3.12 – Comparaison entre les résultats du modèle (en bleu) et les données réelles (en orange)

On remarque que notre modèle est un peu trop optimiste sur la production électrique. Sur la figure 3.12, les 60 premières valeurs retrouvées par notre modèle sont au-dessus des valeurs de production réelles. Lorsque la production est supérieure à 155 MW/h, notre modèle est fidèle à la production réelle. En analysant la figure 3.13, on remarque qu'il y a une sur-estimation de la production pour des débits inférieurs à $650\,m^3/s$, mais que pour des débits supérieurs, l'estimation est correcte.

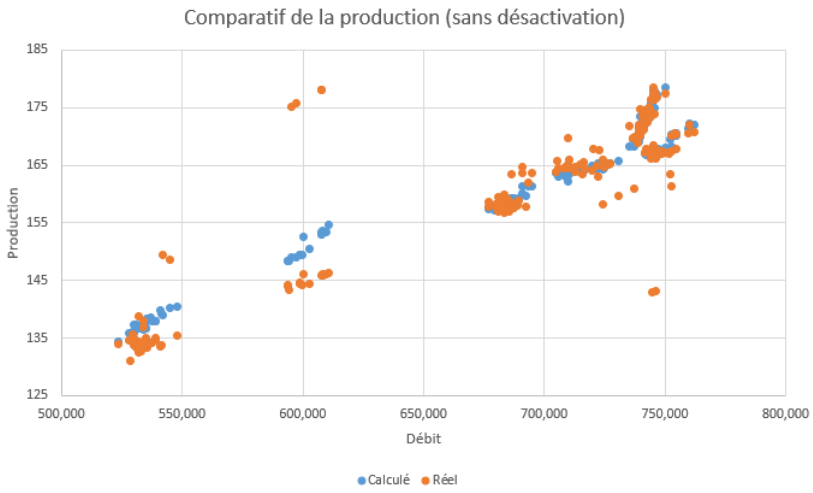

FIGURE 3.13 – Comparaison entre les résultats du modèle (en bleu) et les données réelles (en orange) en fonction du débit

3.6.2 Test avec maintenance

Nous avons obtenu le résultat suivant lorsque nous avons supposé qu'un 0 dans les données signifiait que la turbine était en maintenance.

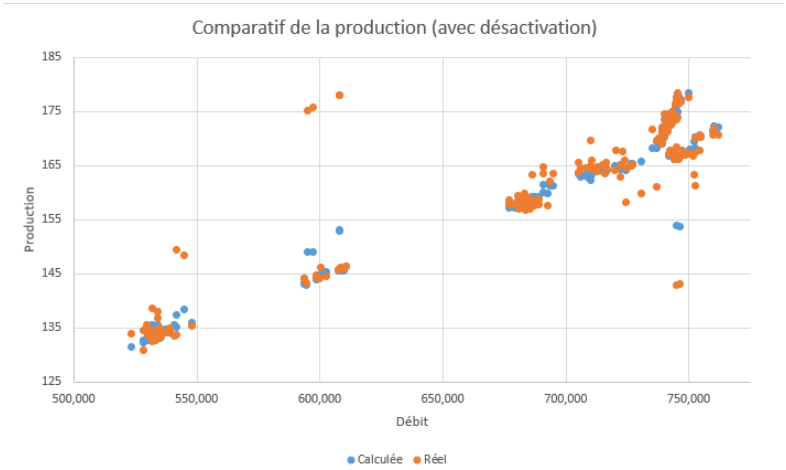

FIGURE 3.15 – Comparaison entre les résultats du modèle (en bleu) et les données réelles (en orange) en fonction du débit

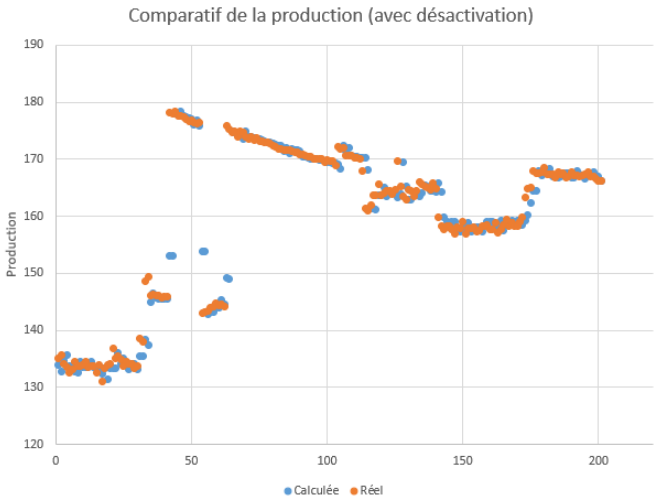

FIGURE 3.14 – Comparaison entre les résultats du modèle (en bleu) et les données réelles (en orange)

En prenant en compte la maintenance, on remarque que les résultats de la figure 3.14 sont très proches de la réalité. De même, lorsqu'on regarde la figure 3.15, les valeurs calculées sont plus proches de la réalité. Cependant, il y a parfois de grandes disparités, notamment pour $600m^3/s$ où notre modèle n'arrive pas à atteindre les valeurs de production optimales.

3.7 Performances

Afin de calculer le temps réel de calcul (car notre programme lit et écrit dans des fichiers pour les tests), nous avons fait une somme du temps de chaque simulation et nous avons divisé par le nombre total de simulations. En moyenne, une simulation prend $6,441403ms$. Le programme au complet utilise entre 700 et 800 Mo lorsqu'on simule les 200 lignes.

Cette expérience a été réalisée sur une JVM non modifiée, et sur un PC disposant d'un processeur Intel Core i5-7300HQ 2.50GHz et de 8Go de RAM.

3.8 Retours sur la partie 1

Après analyse de nos résultats, on se rend compte que la turbine 2 et 5 sont sûrement mal modélisées. En effet, notre résolution par programmation dynamique ne va pas chercher les meilleures valeurs de la turbine 5 (rapport d'efficacité) et utilise la turbine 2 comme buffer. Ce qui signifie que les valeurs choisies sont les plus efficaces pour notre programme, car il ne met jamais de turbine à 0 alors qu'en réalité, c'est la chose à faire pour chercher les valeurs maximales. Nos modélisations de turbines ne sont donc pas si proches de la réalité. Ce qui est étonnant, c'est que notre production totale en fonction du débit à répartir est lui proche de la réalité. En utilisant une discrétisassions plus fine (3), on se rend compte que nos meilleurs modèles sont les turbines 3 et 4. Les turbines 1,2 et 5 commencent à se comporter comme la turbine 2. Nous savions que la turbine 5 était la moins bien modélisée lors de la partie 1 (nous n'avons pas réussi à trouver de fonction qui se rapproche de son comportement). Nous pensons également que la différence de résultats s'explique par l'accumulation d'approximation, car en plus des fonctions de productions, nous avons approximé l'élévation aval en fonction du débit turbiné et de l'élévation amont. Ce qui, en plus de la discrétisassions, augmente les marges d'erreur.

Discussions

Nous pensons que nous avons correctement implémenté l'algorithme de programmation dynamique, mais que nous souffrons des modélisations réalisées en première partie. En effet, via la programmation dynamique, nous attribuons les meilleures valeurs possibles pour un débit donné, et ce n'est pas, comme nous le craignons, la première turbine qui est au maximum, la seconde également, .., puis la dernière avec le reste de débit non attribué. C'est une répartition qui se fait selon les fonctions de productions des turbines. Hors, avec nos résultats, on se doute qu'il faudrait retravailler ces équations pour améliorer la modélisation. Nous avions pris en compte ce problème en fin de partie 1, et avons bien architecturé notre programme afin de modifier facilement les fonctions de production de chaque turbine et ainsi tenter d'améliorer le modèle.
Concernant la programmation dynamique, nous n'avons pas eu de soucis particuliers. En effet, une foi que le modèle est posé, il suffit de l'implémenter. En tirant parti du polymorphisme des langages orientés objets, l'implémentation se résume à un algorithme générique. Nous sommes cependant satisfait des répartitions proposés par notre modèle, qui permettent de d'obtenir une répartition semblable à la réalité, et des productions en fonction du débit également semblable. Nous aurions aimé que notre programme aille chercher au maximum l'efficacité maximale des turbines, en obligeant une turbine à ne rien produire.

Conclusion

Dans la partie I, nous avons réussi à modéliser les différentes fonctions de production grâce à Matlab et à sa boîte à outils "Curve Fitting". Les turbines 1, 2, 3 et 4 possèdent des modèles similaires car se sont les quatre turbines d'origine. Les coefficients des équations modélisant leur production sont cependant différents car chaque turbine possède sa propre fonction de production (ce ne sont pas parfaitement les mêmes). La turbine 5, installée plus tard, possède elle une fonction de production différente des précédentes. Cette turbine est plus puissante. Nous avons pu voir dans ce rapport, que nos fonctions de productions ont un optimum différent de la réalité, ce qui empêche notre modèle global d'avoir les mêmes résultats. De même, les fonctions de production sont souvent légèrement optimistes par rapport aux fonctions de productions réelles. Cependant, nos résultats finaux concordent avec le modèle réel. Il est également nécessaire, à l'heure actuelle, d'obliger le système à mettre une turbine à 0. En effet, à cause des modélisations des fonctions de productions qui ne sont pas assez proches des fonctions de productions réelles, le système ne met jamais une turbine à 0 par lui-même. Cela reste cependant cohérent, car notre modèle trouve une production totale supérieure lorsqu'on ne l'oblige pas à mettre une turbine à 0 que lorsqu'on l'oblige (via le postulat de maintenance).
Nous allons, pour la suite de ce projet, nous concentrer sur l'élaboration d'une interface graphique, qui permettra à un opérateur de lancer facilement la simulation, afin de répartir au mieux le débit et ainsi maximiser la production du barrage des Chutes du Diable. Nous allons également tenter de remodéliser les fonctions de production des turbines 2 et 5.