Optimisation de la production d'une centrale hydroélectrique
Projet de session 2

Hebert Ségolène, Seuanes-Pereira Jean-Baptiste, Pattier Lilian

Hebs02629605, Seuj18099601, Patl21039604

8 avril 2019

Maîtrise informatique (3037)

Cours 8INF912 - Résolution de problèmes industriels

Table des matières

Table des figures

Introduction

Contexte d'étude

L'hydroélectricité constitue, au Québec, la première source d'énergie. Il est donc naturel que les sujets d'étude, notamment dans le domaine de l'optimisation, fleurissent dans ce domaine, dans cette région.

Dans cette étude, nous répondons à l'appel d'offres de l'entreprise X qui est implantée au Saguenay, dans le domaine de l'hydroélectricité, depuis longtemps. Le contrat déposé par cette dite entreprise consiste en le développement d'une solution permettant de gérer la production des turbines dans leurs centrales. Il s'agit bien entendu de maximiser la puissance totale fournie afin de maximiser les gains de l'entreprise.

La centrale à laquelle nous nous intéressons tout particulièrement pour le développement de notre solution de gestion des turbines est la centrale Chute-Savane faisant partie du système Saguenay Lac-Saint-Jean. L'entreprise X nous fournit pour cette centrale les données réelles de production et le débit turbiné de chaque turbine, ainsi que l'élévation amont et le débit total turbiné de 2013 à 2017 pris toutes les 2 minutes.

Rappels des résultats obtenus précédemment

Lors d'une précédente étude, nous avons pu modéliser les fonctions de production de chaque turbine en fonction de la hauteur de chute nette ainsi que du débit turbiné en utilisant les données reçues de l'entreprise X. Une modélisation de l'élévation avale en fonction du débit total turbiné a aussi été réalisée. Pour rappel, ces modélisations ont été obtenues à l'aide de la régression polynomiale. Pour obtenir les coefficients des polynômes modélisant nos fonctions, nous nous étions aussi fixé comme objectif de parvenir à un coefficient de détermination R^2 d'au moins 0.999 et de choisir les degrés les plus bas possibles. Il est à noter que la fonction de production de la turbine 3 a été corrigée étant donné que les données réelles que nous avions reçues pour cette turbine étaient erronées. Les modélisations ainsi obtenues sont les suivantes :

$$f_{elav}(Q_{tot}) = -1,393.10^{-6} + 0,006943Q_{tot} + 100Q_{tot}^2 \tag{1}$$

$$P_1(h,q) = 20,44 - 0,5908h - 0,138q + 0,01402hq - 0,0003412q^2 \tag{2}$$

$$P_2(h,q) = 0,6415 - 0,01847h - 0,1848q + 0,004656hq + 0,003525q^2 + 3,52.10^{-5}hq^2 - 1,655.10^{-5}q^3 \tag{3}$$

$$P_3(h,q) = 0,7799 - 0,02261h + 0,1995q - 0,001695hq - 3,519.10^{-5}q^2 + 7,235.10^{-5}hq^2 - 9,338.10^{-6}q^3 \tag{4}$$

$$P_4(h,q) = 17,91 - 0,512h - 0,4176q + 0,0106hq + 0,004513q^2 + 2,047.10^{-5}hq^2 - 1,827.10^{-5}q^3 \tag{5}$$

$$P_5(h,q) = 16,02 - 0,4612h - 0,08307q + 0,01352hq - 0,0004633q^2 \tag{6}$$

Objectifs

Cette étude vise à exploiter les modélisations présentées ci-dessus afin de créer un algorithme capable de décider du débit à turbiner à associer à chaque turbine afin de maximiser la puissance totale produite. L'algorithme devra s'adapter à la spécification d'un débit maximal pouvant être turbiné pour chaque turbine ainsi qu'à la spécification de la non-utilisation de turbines.

L'algorithme utilisé est un algorithme dynamique. Le fonctionnement général d'un tel algorithme sera décrit. Nous décrirons aussi son utilisation spécifique à notre cas en présentant le modèle mathématique de programmation dynamique. L'implémentation de l'algorithme avec notamment les contraintes de spécification de non-utilisation et de débit maximal pour chaque turbine sera aussi présentée. Finalement, nous présenterons les résultats obtenus, les comparerons aux données fournies par l'entreprise X et discuterons de ces derniers.

1 Programmation dynamique

Les problèmes d'optimisation peuvent être résolus à l'aide de méthodes algorithmique que l'on appelle programmation dynamique [1]. Cela consiste à décomposer le problème initial en plusieurs sous-problèmes qui, eux-mêmes, peuvent aussi être décomposés en plusieurs sous-problèmes et ainsi de suite, comme on peut le voir sur la figure suivante :

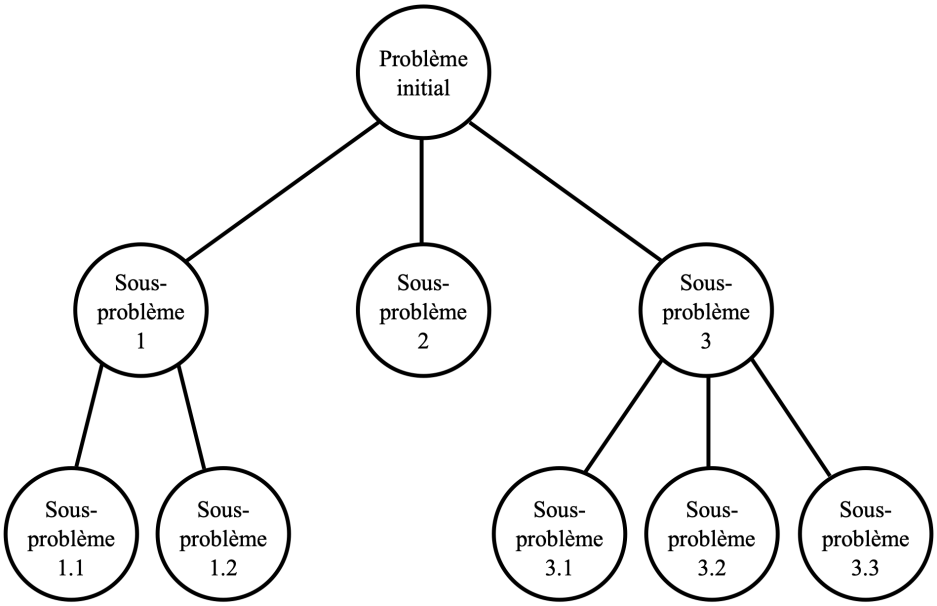


FIGURE 1 – Décomposition d'un problème en plusieurs sous-problèmes

On choisit ensuite de résoudre des problèmes dans un ordre croissant, tout en stockant les résultats intermédiaires. En effet, plus les sous-problèmes sont petits, plus il est facile de les résoudre. C'est d'ailleurs l'intérêt principal de cette méthode algorithmique. Une fois que l'on a obtenu les solutions optimales de chacun des sous-problèmes qu'on a résolu, on construit la solution globale qui sera alors obligatoirement optimale, d'après le principe de Bellman.

Usuellement, il est plus facile de résoudre les problèmes d'optimisation avec la programmation dynamique lorsqu'ils ont une structure en étapes [2]. Sinon, il n'est pas évident de trouver une manière de formuler le problème avec un algorithme de programmation dynamique. Ainsi, les problèmes d'optimisation résolus le plus couramment avec la programmation dynamiques sont le problème du plus court chemin, l'ordonnancement, le problème du sac à dos, le problème de partitionnement, les problèmes de production ou d'inventaires ou encore l'hydroélectricité. C'est pourquoi nous utilisons cette méthode pour résoudre notre projet et répondre à l'appel d'offres de l'entreprise X. Un autre problème très connu est le calcul de la suite de Fibonacci qui est très efficace et peu coûteuse avec un algorithme de programmation dynamique.

Comme nous l'avons vu précédemment, on décompose notre problème en plusieurs sous-problèmes que l'on appelle étape [2]. Chacune de ces étapes comporte alors des états. De plus, à chaque étape, une décision est prise qui affectera alors l'état de la prochaine étape. On définit également une relation récursive entre chaque étape nous permettant ainsi de calculer la solution optimale. Cela peut alors se représenter de la manière suivante, avec x_n la variable de décision et f la fonction récursive :

On pose alors la fonction récursive (linéaire ou non) suivante :

$$f_n(s_n, x_n) = f^*_{n+1}(s_{n+1}) + g_n(s_n, x_n)$$

Ainsi, à chaque étape, on cherche à minimiser (ou maximiser) le coût suivant :

$$f^*_n(s_n) = \min f_n(s_n, x_n)$$

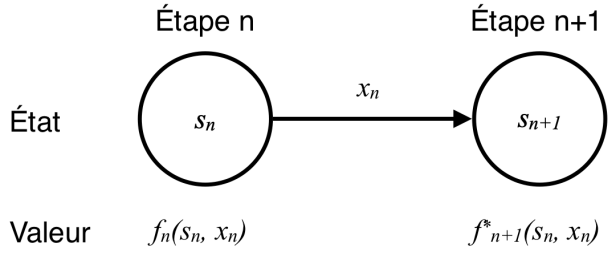


FIGURE 2 – Résolution d'un problème d'optimisation avec la programmation dynamique

La programmation dynamique comporte deux phases : le backward pass et le forward pass. Le backward pass débute à la dernière étape et recule d'une étape tant que toutes les étapes ne sont pas résolues. À l'inverse, la forward pass avance d'étape en étape pour construire la solution optimale. Ainsi, le backward pass débute à l'étape n et considère toutes les possibilités. On peut alors les représenter par le tableau suivant :

s_n	$f_n^*(s_n)$	x_n^*
0	0	0
1	$g_n(1)$	1
2	$g_n(2)$	2
...	$g_n(...)$	...
s_n^{max}	$g_n(s_n^{max})$	s_n^{max}

TABLE 1 – Backward pass à l'étape $n = N$

Ensuite, on continue de construire le backward pass pour toutes les étapes $n \in \{2, \ldots, N-1\}$. On obtient alors le tableau de résultats suivant :

$s_n \backslash x_n$	0	1	...	x_n^{max}	$f_n^*(s_n)$	x_n^*
0	$f_n(s_n, x_n)$	$f_n(s_n, x_n)$	...	$f_n(s_n, x_n)$	$f_n^*(s_n)$	x_n^*
1	$f_n(s_n, x_n)$	$f_n(s_n, x_n)$	...	$f_n(s_n, x_n)$	$f_n^*(s_n)$	x_n^*
2	$f_n(s_n, x_n)$	$f_n(s_n, x_n)$	...	$f_n(s_n, x_n)$	$f_n^*(s_n)$	x_n^*
...	...	...	...	...	...	...
s_n^{max}	$f_n(s_n, x_n)$	$f_n(s_n, x_n)$	...	$f_n(s_n, x_n)$	$f_n^*(s_n)$	x_n^*

TABLE 2 – Backward pass à l'étape $n \in \{2, \ldots, N-1\}$

Enfin, pour la première étape ($n = 1$), on termine le backward pass avec le tableau suivant :

$s_n \backslash x_n$	0	1	...	x_n^{max}	$f_n^*(s_n)$	x_n^*
0	$f_n(s_n, x_n)$	$f_n(s_n, x_n)$	...	$f_n(s_n, x_n)$	$f_n^*(s_n)$	x_n^*

TABLE 3 – Backward pass à l'étape $n = 1$

Ainsi, le backward pass nous a permit de construire l'ensemble des tableaux de résultats de la programmation dynamique. Nous n'avons plus qu'à construire la solution optimale à l'aide du forward pass qui évoluera alors de $n = 1$ à $n = N$. On localisera alors la solution optimale pour chaque étape grâce au tableau de backward pass correspondant et on remonte alors toutes les étapes jusqu'à l'étape N. On a alors construit la solution optimale du problème.

La programmation dynamique est donc déterministe. En effet, il n'y a aucune incertitude concernant les paramètres. Ainsi, la programmation dynamique stochastique est très utilisée pour les problèmes avec des incertitudes. Cependant, il peu y avoir des problème de dimensionnalité pour les problèmes comportant beaucoup de variables ou de discrétisations.

2 Méthodologie

2.1 Théorie

Le problème consiste à assigner le débit à chaque groupe turbo-alternateurs de la centrale de façon à maximiser la puissance totale.

On note $I = \{1, 2, 3, 4, 5\}$ l'ensemble en bijection avec les groupes turbo-alternateurs de la centrale. On associe donc chaque turbine avec un entier $n \in I$.

Pour un entier positif k appelé discrétisation, on note $J_k = \{ik, i \in \mathbb{N}\}$.

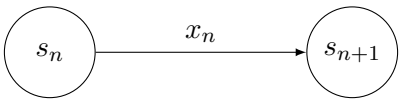
Chaque turbine dispose d'une fonction puissance qui dépend du débit alloué et de la hauteur de chute nette. On note pour $n \in I$, P_n la fonction puissance associée à la turbine n. On note x_{tot} le débit total à allouer.

2.1.1 Cas de base

Pour simplifier l'explication, on considère d'abord le cas ou toutes les turbines sont actives, puis nous expliquerons notre démarche dans le cas ou certaines turbines ne le sont pas.

Caractérisation du problème :

- Les étapes sont les différentes turbines $n \in I$. Il y'a donc 5 étapes en tout.
- Pour $n \in I$, la variable $x_n \in J_k$ correspond au débit alloué à la turbine n.
- Pour $n \in I$, l'état $s_n \in J_k$ correspond au débit restant à allouer à l'étape n.
- On cherche à maximiser la quantité $\sum_{n=1}^{5} P_n(x_n)$ sous les contraintes $\sum_{n=1}^{5} x_n = x_{tot}$ et pour tout $n \in I$, $x_n \leq x_n^{max}$ avec x_n^{max} le débit maximum que l'on peut allouer à la turbine n.



$$s_{n+1} = s_n - x_n$$

$$f_n(x_n, s_n) = f_{n+1}^*(s_{n+1}) + g_n(x_n)$$

avec

$$g_n(x_n) = P_n(x_n)$$

$$s_n^{max} = \min \left\{ x_{tot},\ \sum_{i=1}^{n-1} x_i^{max} \right\}$$

NB. Bien sûr, les puissances dépendent également de la hauteur de chute nette, bien que nous ne l'ayons pas précisé pour simplifier l'écriture.

A chaque étape n, le débit restant maximum à allouer s_n^{max} est le minimum entre le débit total à allouer x_{tot}, et la somme des débits maximum allouables aux turbines précédentes $(n-1, n-2, ...1)$. En effet, supposons par exemple que nous sommes à l'étape 3 et que chaque groupe peux turbiner $10m^3/s$ au maximum, il ne serait alors pas pertinent de noter $s_3^{max} = 40$, cela voudrait dire que même si les turbines tournaient à plein régime, il resterait à la fin du problème $10m^3/s$ à allouer, ce qui rendrait le problème non optimal. De plus, dans ce même cas de figure, si le débit total à allouer était de $20m^3/s$, il ne serait pas pertinent d'écrire $s_3^{max} = 30$, cela signifierait in fine que nous autoriserions les groupes à turbiner plus de débit que disponible.

Exemple simplifié :

Discrétisation : $k = 10$. On dispose de 3 turbines dont les puissances en fonction du débit sont données par le tableau suivant :

	P_1	P_2	P_3
0	0	0	0
10	8	5	8
20	12	8	10
30	14	11	12
40	16	14	14
50	19	17	16

Résolution :

s_3	$f_3^*(s_3)$	x_3^*
0	0	0
10	8	10
20	10	20
30	12	30
40	14	40
50	16	50

TABLE 4 – Étape $n = 3$

$s_2 \backslash x_2$	0	10	20	30	40	50	$f_2^*(s_2)$	x_2^*
0	0	-	-	-	-	-	0	0
10	$0 + 8 = 8$	$5 + 0 = 5$	-	-	-	-	8	0
20	$0 + 10 = 10$	$5 + 8 = 13$	$8 + 0 = 8$	-	-	-	13	10
30	$0 + 12 = 12$	$5 + 10 = 15$	$8 + 8 = 16$	$11 + 0 = 11$	-	-	16	20
40	$0 + 14 = 14$	$5 + 12 = 17$	$8 + 10 = 18$	$11 + 8 = 19$	$14 + 0 = 14$	-	19	30
50	$0 + 16 = 16$	$5 + 14 = 19$	$8 + 12 = 20$	$11 + 10 = 21$	$14 + 8 = 22$	$17 + 0 = 17$	22	40
60	-	$5 + 16 = 21$	$8 + 14 = 22$	$11 + 12 = 23$	$14 + 10 = 24$	$17 + 8 = 25$	25	50
70	-	-	$8 + 16 = 24$	$11 + 14 = 25$	$14 + 12 = 26$	$17 + 10 = 27$	27	50

TABLE 5 – Étape $n = 2$

$s_1 \backslash x_1$	0	10	20	30	40	50	$f_1^*(s_1)$	x_1^*
70	$0 + 27 = 27$	$8 + 25 = 33$	$12 + 22 = 34$	$14 + 19 = 33$	$16 + 16 = 32$	$19 + 13 = 32$	34	20

TABLE 6 – Étape $n = 1$

Politique optimale :

$$x_1^* = 20$$

$$x_2^* = 40$$

$$x_3^* = 10$$

2.1.2 Prise en compte des turbines hors-service

Dans le cas ou une ou plusieurs turbines seraient inactives, on réassigne les k turbines actives restantes dans l'ensemble $I_k = \{1, ..., k\} \subset I$ correspondant puis on applique la stratégie précédente.

2.1.3 Débit total supérieur au débit supporté par les turbines

Dans le cas où le débit total est supérieur à la somme des débits max que peuvent turbiner les groupes actifs ($x_{tot} > \sum_{n \in I_k} x_n^{max}$ dans le cas de k turbines en service), on divise le débit total en deux tel que :

$$x_{tot} = x_{turb} + x_{van}$$

avec

$$x_{turb} = \sum_{n \in I_k} x_n^{max}$$

Dans ce cas de figure, les turbines actives tourneront toute à plein régime. Cependant, le débit évacué fait également varier la hauteur de chute nette. La production diffère alors dans le cas où $x_{van} = 0$ et $x_{van} > 0$, bien que les turbines tournent à plein régime dans les deux cas. Nous prendrons donc cela en compte dans notre implémentation pratique.

2.2 Pratique

Nous avons implémenté notre programme avec le langage JAVA en respectant le paradigme orienté objet. Nous n'avons pas utilisé de pattern d'architecture particulier, mais nous avons regroupé les différentes classes en 3 packages suivant leur finalité (principal, elements, algorithme). Le projet est subdivisé en plusieurs classes :

- Classe `Centrale` : contient toutes les informations relatives à la centrale comme les composants structurels (Liste d'objet Turbines pour le parc des groupes turbo-alternateur) ainsi que les différentes données inhérentes.
- Classe `Turbine` : contient toutes les informations relatives à une turbine (fonction de puissance, hauteur de chute nette, état de fonctionnement...)
- Classe `Polynome` : permet d'implémenter un polynômes univarié ou multivarié (surcharge du constructeur) ainsi que l'évaluation en fonction de valeurs données. Cette classe permet de définir les différentes fonctions puissance associées aux turbines.
- Classe `Constructeur` : permet d'instancier la centrale en fonction des différents paramètres associés (parc de turbines, débit à turbiner...)
- Classe `ProgrammationDynamique` : permet d'instancier le problème mathématique en fonction d'une centrale et des paramètres donnés (débit à turbiné, discrétisation) et également de le résoudre. Ainsi, le programme dynamique de résolution est implémenté dans cette classe à l'aide de plusieurs méthodes : créations des différents tableaux, fonction de forward pass...

3 Résultats

Nous avons testé notre simulation sur les 100 première lignes du fichier de données Excel de façon à comparer nos simulations avec les valeurs réelles. À cet effet, nous avons créé une classe permettant de lire et écrire dans des fichiers Excel. Puis à l'aide d'une boucle, nous avons itéré sur chaque ligne de la feuille de référence, effectué la simulation pour les valeurs de débit total et d'élévation amont correspondante (tout en prenant garde de désactiver les turbines lorsqu'elles le sont dans le fichier initial), puis écrit les résultats de la simulation dans une nouvelle feuille Excel. Pour s'approcher au mieux des résultats réels, nous avons choisi une discrétisation du débit de 1 m^3/s.

Nous obtenons ainsi les résultats suivants :

	Simulation	Données réelles	Erreur relative	Corrélation
Elev. avale moyenne	103.56	103.78	-0.22	0.958
Puissance moyenne	181.09	180.14	0.95	0.999

TABLE 7 – Résultats de la simulation : centrale

Moyenne	Débit simulation	Débit réel	Puissance simulation	Puissance réelle
Turbine 1	83.81	99.73	25.31	29.32
Turbine 2	132.40	119.35	41.13	37.15
Turbine 3	86.72	84.05	25.98	25.26
Turbine 4	151.83	139.15	48.28	44.53
Turbine 5	125.02	138.50	40.38	43.87

TABLE 8 – Résultats de la simulation : turbines

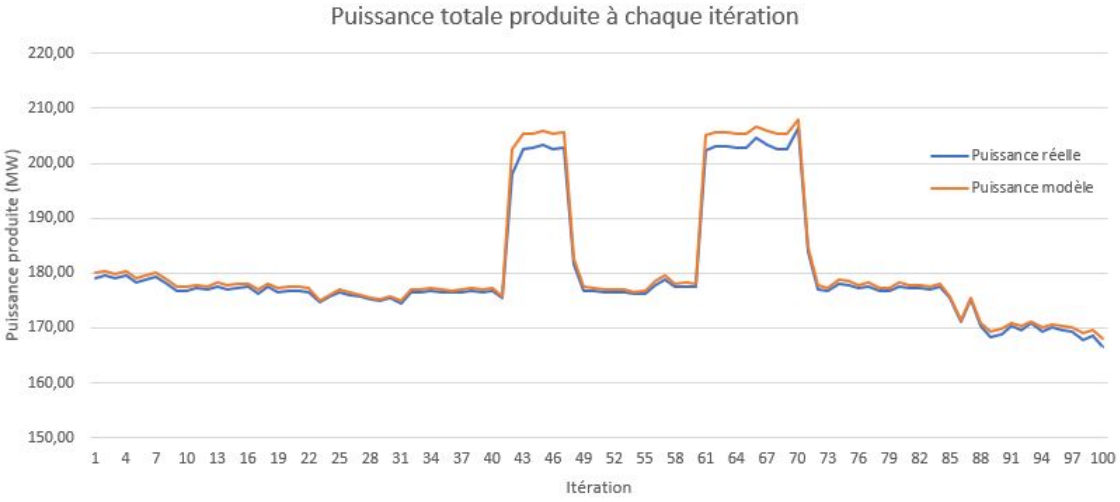


FIGURE 3 – Puissance totale produite en fonction de l'itération

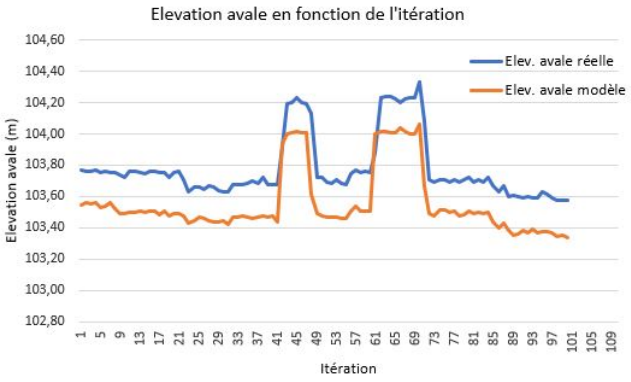


FIGURE 4 – Elevation avale en fonction de l'itération

4 Discussion

Les résultats que nous obtenons sont plutôt satisfaisants. Nous obtenons une puissance moyenne de $181.09MW$ contre $180.14MW$ pour les données réelles. Le coefficient de corrélation entre les valeurs de puissances pour chaque itération est de 0.999 ce qui confirme la validité de notre modèle. La superposition des courbes de la figure 3 renforce cette conclusion.

La répartition des débits entre chaque turbine et les puissances fournies par le modèle et les données

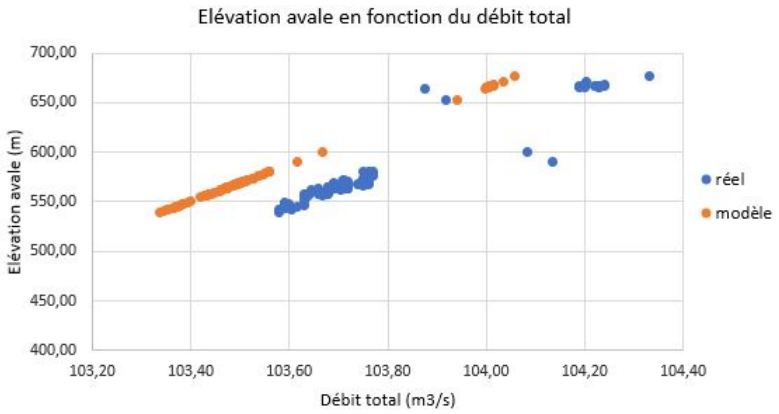


FIGURE 5 – Elevation avale en fonction du débit total

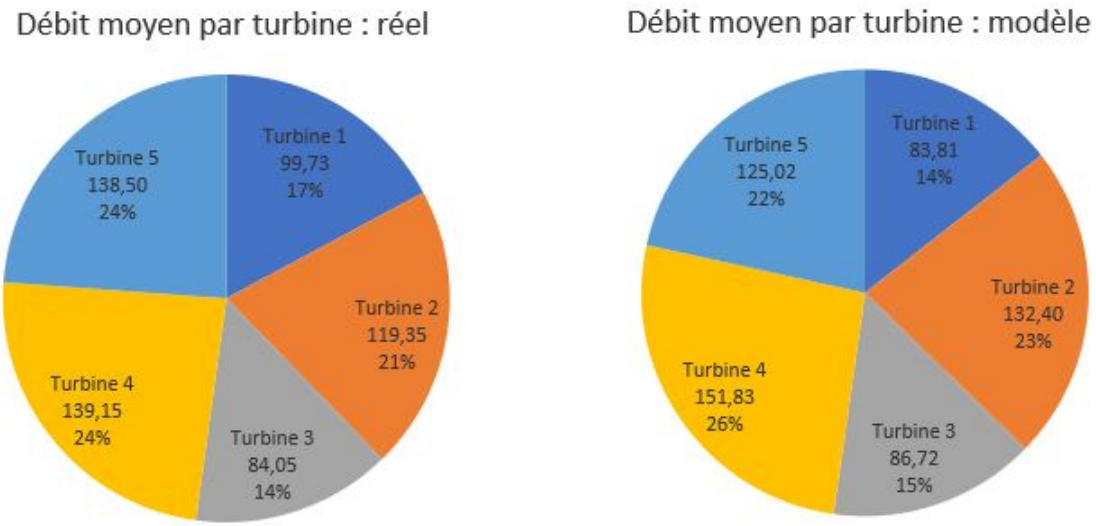


FIGURE 6 – Débits (m^3/s) moyens par turbine

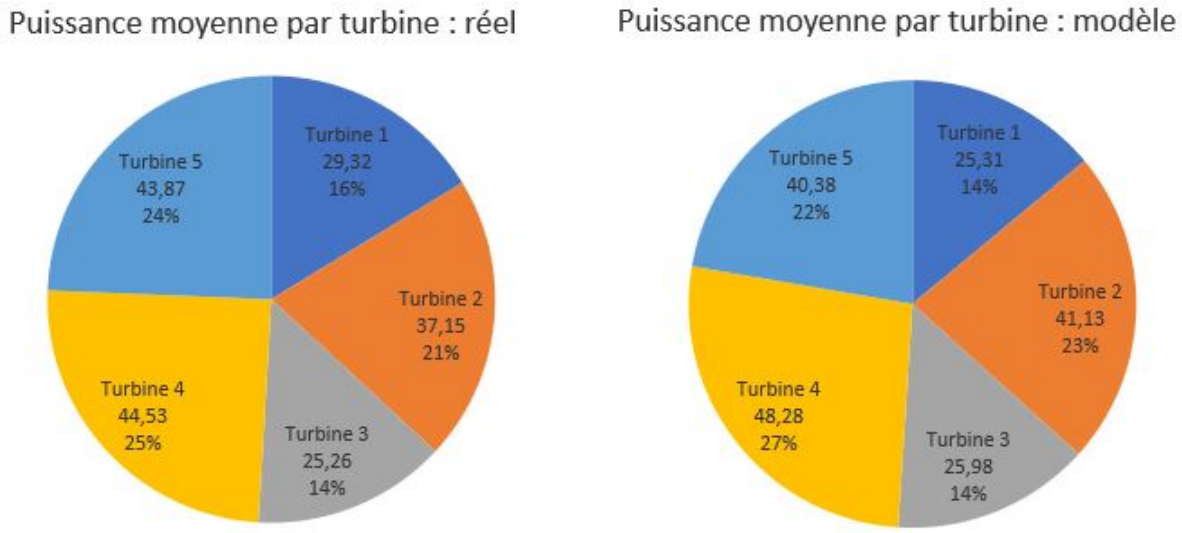


FIGURE 7 – Puissances (MW) moyennes par turbine

réelles sont également corrélées comme nous pouvons le voir sur les figures 6 et 7.

Le principal écart significatif entre les données réelles et notre modèle se situe dans les valeurs d'élévation avale. L'élévation avale est en moyenne $22cm$ plus basse dans notre modèle que pour les données réelles. On distingue notamment cet écart sur la figure 4. Cet écart peut s'expliquer par le fait que l'élévation avale en réalité ne dépend pas uniquement du débit total, alors que nous avons modélisé cette fonction par un polynôme univarié uniquement dépendant du débit total. Nous pouvons constater cela sur la figure 8 présentée dans le dernier rapport où l'amplitude du nuage de points autour de la courbe représentative de notre fonction d'élévation totale est parfois importante. Si nous extrayons de ce graphique uniquement les points correspondant au 100 première données du fichier Excel (figure 9), nous pouvons effectivement voir que les données sont relativement éloignées de notre courbe modèle. Cette différence d'élévation impacte légèrement notre simulation, ce qui peut en partie expliquer la légère supériorité de puissance obtenue par notre modèle comparé aux données réelles.

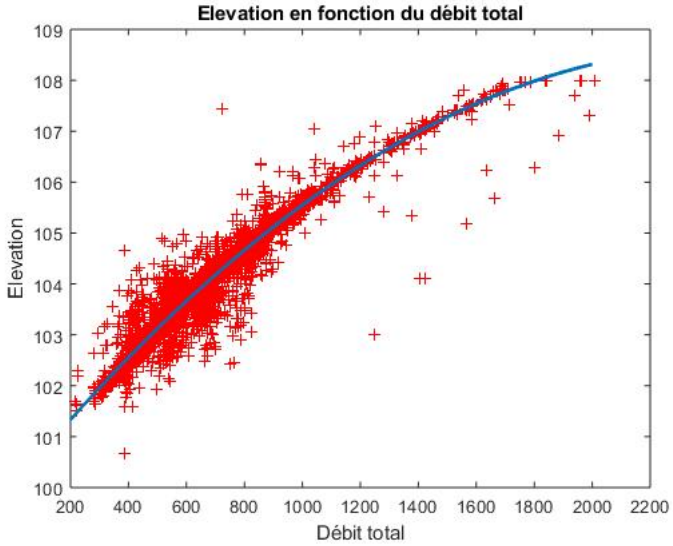


Figure 8 – Modèle d'élévation avale en fonction du débit

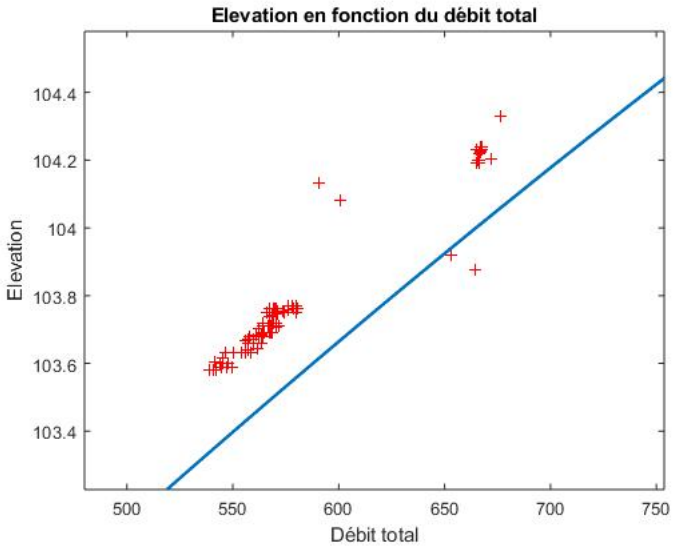


Figure 9 – Comparaison modèle élévation avale sur les 100 première données

Conclusion

Nous avons implémenté un algorithme dynamique capable de prendre en compte des spécifications de non-utilisation de turbines et de débit maximal à turbiner afin de distribuer un débit sur chaque turbine de sorte à maximiser la production d'énergie. Cet algorithme utilise pour étape chaque turbine. Pour cette dernière, on définit comme variable le débit à lui allouer et comme état, le débit lui restant à allouer résultant des étapes précédentes. Pour chaque étape, l'algorithme dynamique trouvera le débit optimal à assigner à la turbine (variable) pour chaque état possible.

Les résultats recueillis par cet algorithme sont assez convaincants. En effet, on retrouve des résultats très similaires, et même un peu supérieurs à ceux que la centrale obtient actuellement. Cela laisse à penser que la gestion des turbines est déjà optimisée avec une technique relativement similaire.

Les perspectives d'améliorations de notre travail sont multiples. Lors d'une prochaine étude, on s'attardera sur la présentation d'une interface graphique (figure 10) afin de faciliter l'emploi de notre algorithme de gestion des turbines de centrales. D'autres travaux pour l'avenir pourraient consister en l'optimisation de la distribution du débit non pas à un temps donné, mais sur une période de temps discrétisée d'une journée par exemple.

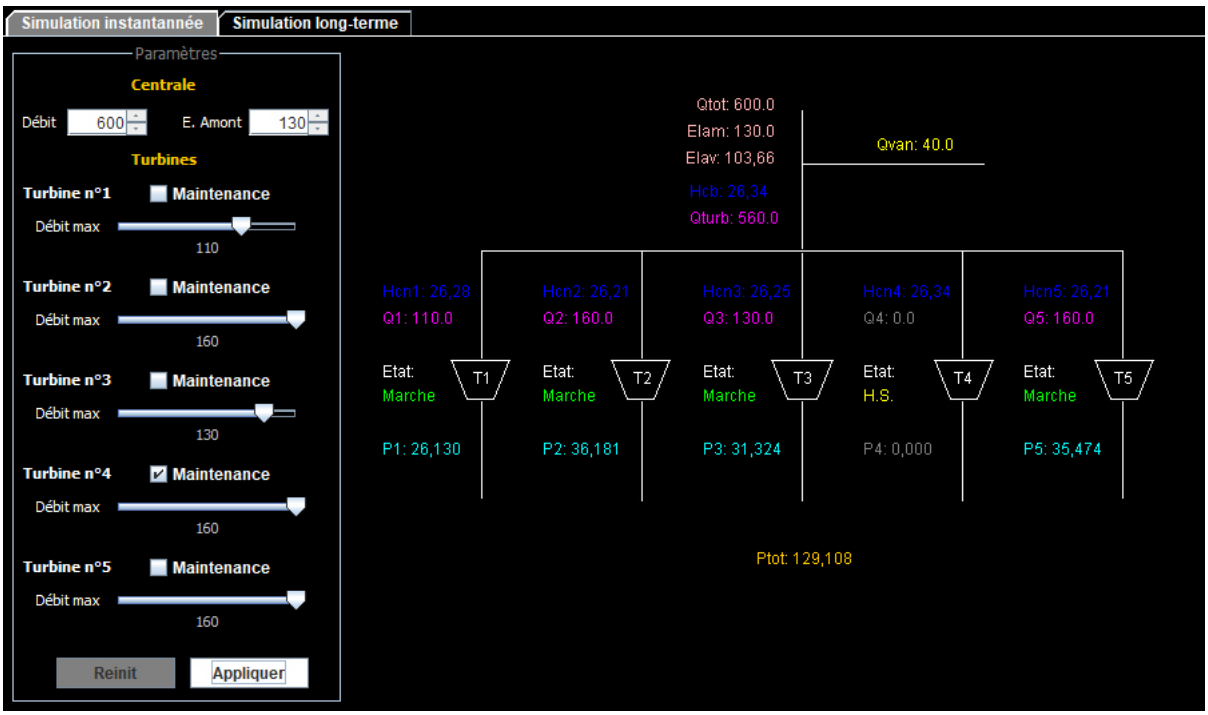


FIGURE 10 – Aperçu de notre interface graphique

Références

[1] Contributeurs de Wikipédia. (2019, 26 mars) *Programmation Dynamique - Wikipédia, l'encyclopédie libre* [Internet]. Disponible à : `https://fr.wikipedia.org/wiki/Programmation_dynamique`

[2] S, Séguin. "Résolution de problèmes industriels - Cours 9 : Programmation dynamique", présenté à l'Université du Québec à Chicoutimi dans le cours Résolution de Problèmes Industriels 8INF912, 2019.