Université du Québec à Chicoutimi

Maîtrise en informatique

Optimisation (8INF914)

Optimisation de la production d'une centrale hydroélectrique partie 2

Auteur :

Geoffrey Glangine

(GLAG01029406)

Bouwendmanegre Jean

Nikiema(NIKB27117501)

Professeur :

Sara Séguin

28 mars 2018

Table des matières

Chapitre 1

Introduction

Dans la première partie de cette étude, nous avons reçu toutes les données de débit turbiné collectées par une centrale hydroélectrique heure par heure pendant 5 ans (2013 à 2017). Ces données nous ont permis de pouvoir modéliser les fonctions de production de chacune des turbines afin de pouvoir réaliser un algorithme de répartition de la charge pour chaque turbine afin d'optimiser la production d'électricité. En effet dans une centrale électrique, chaque groupe turboalternateur possède un certain rendement, et celui-ci est différent sur toutes les turbines à cause de plusieurs facteurs telles que ses composantes mécaniques. Ainsi, nous avons pu valider cette théorie après avoir modélisé les 5 fonctions de productions(rappel en Annexe A). Elles sont toutes différentes, il existe donc une répartition optimale du débit qui fournira une puissance optimale. Pour cette partie du projet, nous allons donc développer un algorithme de programmation dynamique qui nous permettra de trouver cette combinaison optimale de répartition de débit entre les 5 différentes turbines. Dans la partie précédente, nous avons aussi dû modéliser l'élévation aval en fonction du débit total turbiné et de l'élévation amont. Cette fonction nous sera utile dans cette partie pour calculer la hauteur de chute nette qui est nécessaire dans le calcul de la puissance produite par turbine.

Chapitre 2

Programmation dynamique

Pour ce projet, nous avons choisi d'utiliser la programmation dynamique, car elle semble très adaptée pour résoudre ce genre de problèmes, car on peut diviser le problème en étapes. La programmation dynamique est une méthode utilisée en optimisation permettant de calculer une solution optimale en subdivisant le problème en plusieurs sous-problèmes afin de construire la solution optimale.

2.1 Fonctionnement

La programmation dynamique est fondée sur l'optimalité de Bellman. C'est-à-dire que toute politique optimale est composée de sous-politiques optimales où toute partie d'une solution optimale est elle-même optimale. Par exemple dans un problème de plus court chemin reliant une ville x à une ville z alors si le chemin passe par la ville y le chemin entre x et y est optimal. La programmation dynamique s'applique à plusieurs problèmes connus telle que les problèmes de sac à dos, ordonnancement, plus court chemin, etc.

Un algorithme de programmation dynamique va toujours résoudre les sous-problèmes un à un et pour résoudre un sous-problème, il va s'appuyer sur les solutions trouvées dans le sous-problème précédent. Grâce à ce système, il est garanti que les solutions trouvées par l'algorithme de programmation dynamique soient optimales.

La programmation dynamique permet de réduire la complexité spatiale et temporelle par le fait que chaque sous-problème n'est résolu qu'une seule fois. Dans notre cas d'optimisation de la production hydroélectrique, une turbine est un sous-problème. En effet, il aurait été facile de lister toutes les solutions possibles au problème des turbines avec une simple fonction récursive qui commence à la première turbine et qui, pour chaque débit potentiel, va essayer tous les débits pour chaque autre turbine récursivement. Mais avec cette méthode, on va se retrouver à calculer plusieurs fois la puissance produite par chaque turbine avec un certain débit. C'est pour cela qu'en programmation dynamique, on va trouver une solution plus efficace et donc mémoriser les calculs dans des tables de programmation dynamique afin de les réaliser qu'une seule fois. Ensuite, quand on en a besoin, on va chercher dans ces tables les valeurs.

La programmation dynamique se compose de deux phases, la phase avant (forward pass) et la phase arrière (backward pass).

2.1.1 Backward pass

La phase arrière va résoudre tous les sous-problèmes un à un en commençant par le dernier, elle va ensuite remonter jusqu'au premier en reprenant toujours les solutions de l'étape n+1 pour ses calculs.

2.1.2 Forward pass

Une fois que toutes les étapes ont été résolues par le backward pass, on regarde à la première étape quelle combinaison permet de maximiser ou minimiser le problème et on remonte jusqu'à la dernière étape à partir de cette configuration pour construire notre solution finale.

Chapitre 3

Méthodologie

3.1 Modèle mathématique

Pour résoudre ce problème, nous nous sommes d'abord penchés sur la formulation mathématique de celui-ci. Le principe étant de maximiser la production électrique en fonction d'un débit total (Q_{tot}) et d'une élévation amont, on va chercher à maximiser toutes les fonctions de production, on aura donc :

$$max \sum_{i=1}^{5} P_i(Q_i)$$

Avec Q_i le débit alloué à la turbine i.

Et P_i la fonction de production de la turbine i (Annexe A pour rappel).

Ensuite pour les contraintes, on a seulement la contrainte de débit total maximum et la borne minimale de débit :

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

Et :

$$Q_i \geq 0$$

On obtient donc le modèle final :

$$max \sum_{i=1}^{5} P_i(Q_i)$$

S.T.

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

$$Q_i \geq 0$$

Notons que la somme des débits peut être inférieure au débit total alloué, car dans certains cas, il faut déverser de l'eau du réservoir directement si l'on ne veut pas perdre de puissance.

3.1.1 Modèle mathématique avancé

Précédemment, nous avons présenté un modèle de base au problème, mais on peut ajouter des bornes pour ajouter des fonctionnalités au problème comme le fait que les turbines ont un débit maximum. On va donc introduire la variable Q_{max}qui est le débit maximum utilisable par turbine. On aura donc la borne maximum :

$Q_i \leq Q_{max}$

Et on pourrait aller encore plus loin en définissant une borne maximale par turbine, ce qui permettrait d'avoir une borne maximale pour chaque turbine, par exemple si on met la borne maximale d'une turbine à 0 cette turbine ne pourra pas fonctionner pour le calcul. On changera donc la variable Q_{max} par $Qmax_i$ pour obtenir la borne :

$Q_i \leq Qmax_i$

Que l'on ajoutera au modèle présenté dans le paragraphe précédent pour obtenir notre modèle mathématique final.

3.2 Programmation dynamique

Ensuite, après avoir modélisé mathématiquement notre problème, nous avons pu réfléchir sur la programmation dynamique de celui-ci. On a donc déterminé :

— Les étapes

— Les états

— Les variables de décision

Notre but étant de déterminer le débit à allouer par turbine à partir du débit total (Q_{tot}).

Nous avons donc la variable de décision Q_i qui est la quantité de débit à allouer à la turbine i. Chaque turbine est une étape. Les états sont Q_r le débit restant à allouer.

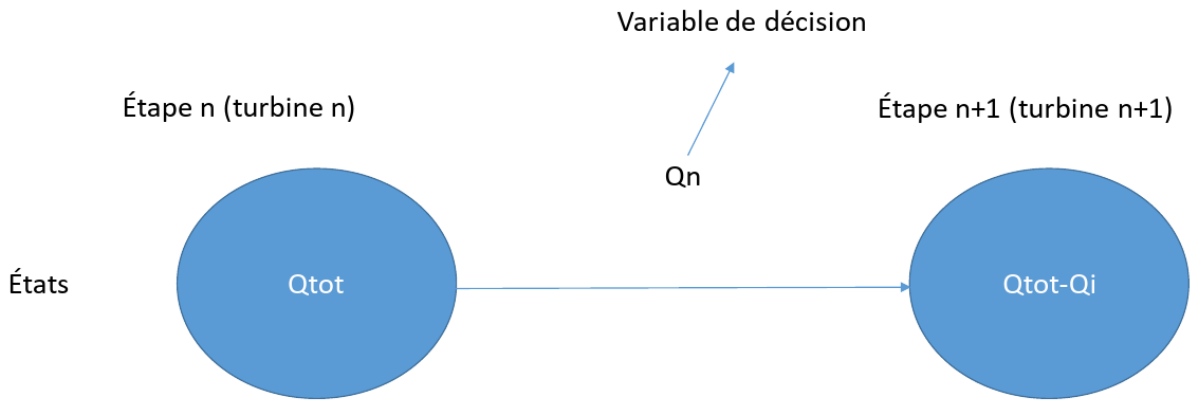

FIGURE 3.1 –

3.3 Technologies utilisées

Pour le langage de programmation, nous avons choisi d'utiliser python 3 car c'est un langage orienté objet qui est simple d'utilisation et très performant pour les calculs mathématiques. Ce langage est beaucoup utilisé dans la recherche mathématique pour ses deux avantages cités plus haut. De plus un de nos membres d'équipe ne maîtrisait pas bien les langages de programmation orienté objet, python s'est présenté être l'un des plus simples à appréhender, nous avons choisi cette solution.

Le logiciel utilisé est quant à lui VIM 8.0 qui est un très bon IDE pour le python avec des plugins de précompilation et d'auto-complétion tels que YouCompleteMe.

Chapitre 4

Résultats

Pour faire cette comparaison de résultats nous avons choisi d'utiliser la discrétisation du débit à $1m^3/s$ car c'est celle qui se rapproche le plus du fichier source. Dans la version finale, la discrétisation du débit affichée sera de $5m^3/s$ car dans la réalité, il n'est pas possible de régler la turbine avec une précision de $1m^3/s$.

Pour se faire, nous avons créé un petit programme python qui prend en entrée des lignes du fichier Excel et qui retourne les différences entre nos résultats fournis par l'algorithme et ce fichier. Le programme calcule aussi l'écart en pourcentage entre la puissance produite par notre algorithme la puissance fournie dans le fichier Excel. Sur l'interface console, on pourra observer à gauche les valeurs calculées par notre algorithme et à droite les valeurs inscrites dans le fichier fourni. La dernière ligne du fichier présente quant à elle l'écart en pourcentage entre la puissance calculée par notre algorithme et la puissance inscrite dans le fichier.

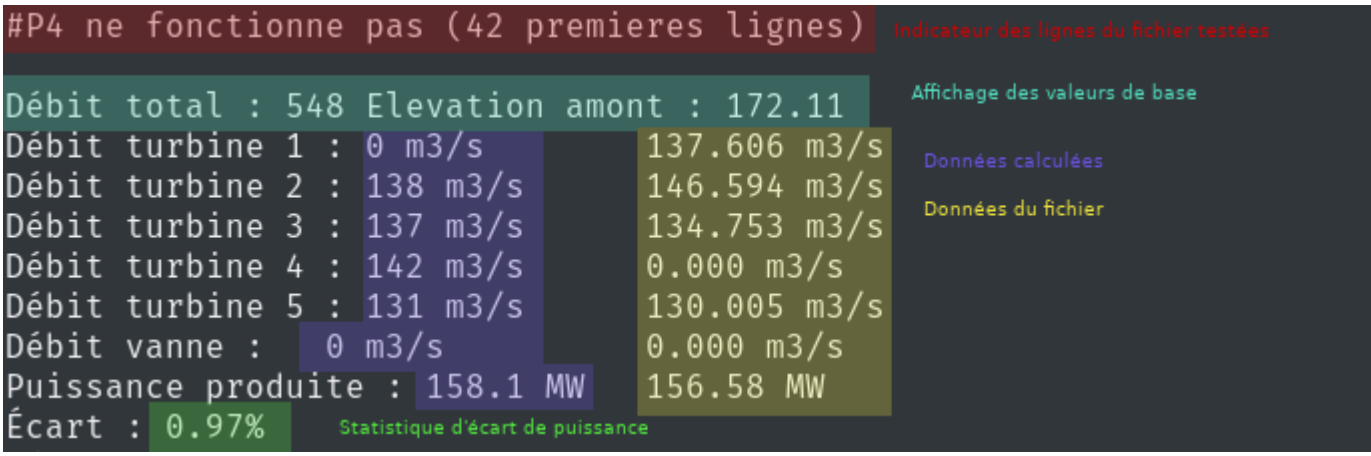

FIGURE 4.1 – Description de l'interface console.

4.1 Tests

Nous avons testé l'algorithme sur les 42 premières lignes, car dans ces lignes-là, la turbine 4 est toujours arrêtée. Nous avons donc eu un affichage identique à celui de la figure 4.1 42 fois, on a donc comparé et regardé manuellement les différences et à l'aide des outils statistiques d'Excel nous avons calculé une médiane et classé les écarts par ordre croissant. Nos résultats sont très proches de la réalité. En effet, on a toujours un écart de moins de 2% entre la puissance totale calculée et celle du fichier. De plus, nous avons toujours la turbine numéro 1 qui est éteinte. La médiane est environ à 1% Voici un autre exemple sur ces 42 premières lignes :

```
Débit total : 546 Elevation amont : 172.12
Débit turbine 1 : 0 m3/s        138.185 m3/s
Débit turbine 2 : 137 m3/s      146.337 m3/s
Débit turbine 3 : 136 m3/s      134.779 m3/s
Débit turbine 4 : 142 m3/s      0.000 m3/s
Débit turbine 5 : 131 m3/s      130.784 m3/s
Débit vanne :   0 m3/s          0.000 m3/s
Puissance produite : 157.63 MW  156.9 MW
Écart : 0.47%
```

FIGURE 4.2 – Exemple sur l'une des 42 premières lignes.

Ensuite, nous avons testé l'algorithme sur les lignes 43 à 48, 62 à 71 et 2569

à 2620 car dans ces lignes-là toutes les turbines sont en marche. Nos résultats sont très bons et on a un écart proche 0.5% dans la plupart des cas. Les écarts maximums sont d'environ 2% sauf pour 2 cas spécifiques où l'on a un écart de puissance de 17% et de 16%. Ce décalage est dû à des erreurs dans le fichier Excel ! En effet, si l'on additionne les débits turbinés par les 5 turbines, on obtient un débit supérieur au débit total turbiné. Ces valeurs sont donc des valeurs aberrantes, on ne va donc pas s'en préoccuper. Globalement, on trouve des résultats meilleurs que sur lest tests avec une turbine éteinte, car la répartition est autour du 0.5% alors que sur la partie avec la turbine éteinte, la répartition était plus hétérogène entre 0 et 2%. Sinon dans tous les autres cas de cette série, les 5 turbines sont actives comme dans le fichier de Rio Tinto.

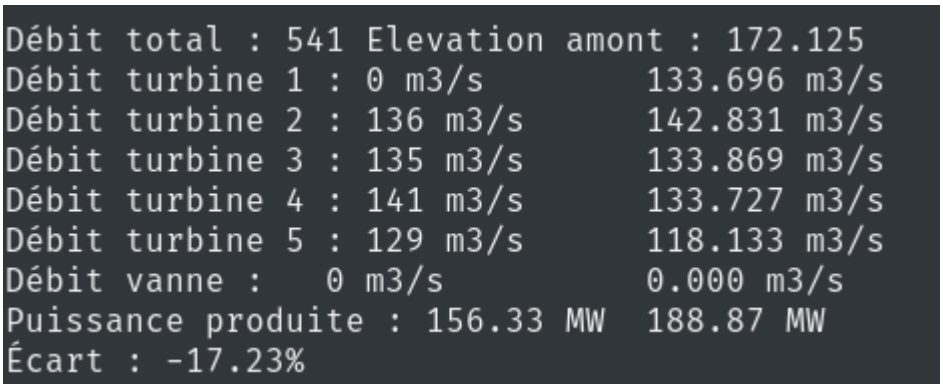

FIGURE 4.3 – Exemple du cas d'erreur isolé.

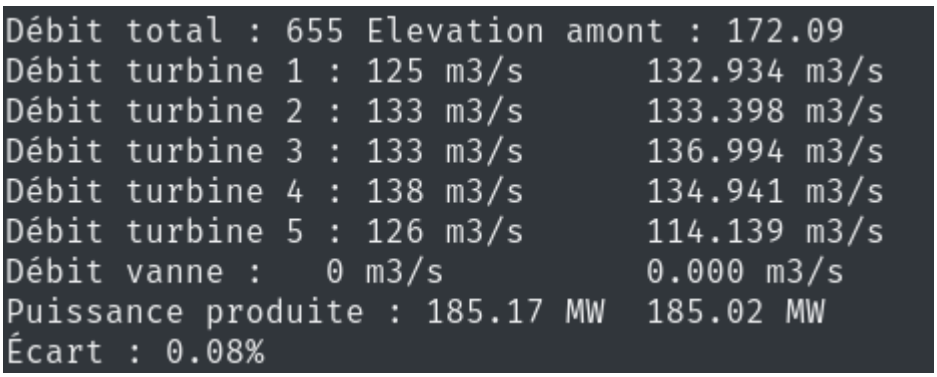

FIGURE 4.4 – Exemple d'un cas avec toutes les turbines actives.

Après nous avons testé les lignes 12591 à 12637 car dans ces lignes du fichier,

la vanne de déversement est ouverte. Pour ce jeu de test, on a effectivement dans notre algorithme toujours toutes les turbines allumées et un déversement dans la vanne. Ici, notre algorithme se comporte encore comme les données que nous avons reçu de Rio Tinto. Cependant, c'est avec ce jeu de données là que nous avons les moins bonnes statistiques. En effet, les écarts de puissances sont situés entre 0.91% et 3.97% et on a 80% des valeurs comprises entre 2% et 3% la médiane est à 2.76%. Mais notre puissance est quand même toujours supérieure à celle du fichier Excel. C'est certainement, car notre déversement par la vanne est toujours moins important que celui de Rio Tinto. Cela est peut-être dû à des restrictions spéciales dont nous n'avons pas les connaissances.

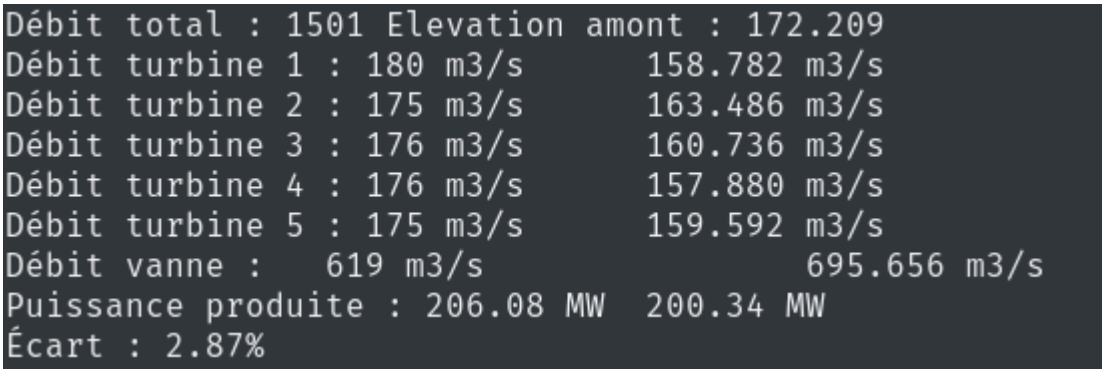

FIGURE 4.5 – Exemple d'un cas avec déversement par la vanne.

Finalement, nous avons testé notre algorithme sur les lignes 33426 à 33475 car dans ces lignes-là, on a du déversement par la vanne et deux turbines sont éteintes. Dans ce cas-là, on a de grosses différences entre notre algorithme et les valeurs reportées par Rio Tinto notre algorithme trouve toujours une puissance plus élevée de plus de 50% que celle relevée par Rio Tinto. On peut réfléchir et se dire que déverser de l'eau par la vanne avec deux turbines éteintes n'est pas très logique. En effet, notre algorithme propose de turbiner avec les 5 groupes et de ne rien déverser. Il y avait peut-être une opération de maintenance sur les groupes hydroélectriques ces jours-là (du 24 au 28 octobre 2016) ? On peut uniquement supposer un facteur de ce type-là (maintenance, panne, restriction quelconque..) car la décision semble être prise de manière non-conforme aux données habituelles

et notre algorithme se comporte très bien sur ce type de cas.

```
Débit total : 765 Elevation amont : 171.625
Débit turbine 1 : 153 m3/s      0.000 m3/s
Débit turbine 2 : 153 m3/s      0.000 m3/s
Débit turbine 3 : 153 m3/s      156.774 m3/s
Débit turbine 4 : 156 m3/s      162.317 m3/s
Débit turbine 5 : 150 m3/s      160.330 m3/s
Débit vanne :   0 m3/s          279.211 m3/s
Puissance produite : 205.75 MW  128.65 MW
Écart : 59.93%
```

FIGURE 4.6 – Exemple d'un cas avec déversement par la vanne avec des turbines éteintes.

4.2 Performance

Dans notre algorithme, il est facile de modifier la discrétisation du débit, et grâce à cela nous avons pu faire des tests de performance. Les tests de performance ont été réalisés sur un ordinateur Dell XPS avec 16GO de ram et un intel core i7 7770HQ(2.8GHz). Sur une discrétisation du débit de $5m^3/s$ on obtient le résultat instantanément. Sur une discrétisation du débit de l'ordre de $1m^3/s$ on obtient un résultat en quelques secondes (environ 2s) si l'on passe la discrétisation à l'ordre de 100 litres/s ($100dm^3/s$) on obtient un résultat au bout de 1 minute 30 et on remarque la complexité spatiale de l'algorithme de programmation dynamique, car on a environ 4Go de ram qui sont utilisés par le programme. Nous avons ensuite testé pour une discrétisation de 10 litres/s, et là, on se retrouve avec le programme qui s'arrête par manque de mémoire vive (16GO disponibles) au bout d'environ 10 minutes.

4.3 importance de la modélisation de la partie 1

Pour tester l'importance de la modélisation de la partie 1, nous avons essayé de lancer l'algorithme avec une autre fonction de modélisation de l'élévation aval en

fonction du débit total, nous avons remarqué qu'en mettant la fonction linéaire :
$Elav = 0.002989 \times Qtot + 137.6$
Que nous avions jugée trop peu précise dans la partie précédente de ce projet par rapport à la fonction polynomiale que nous utilisons actuellement :
$Elav = -7.017 \cdot 10^{-7} \times Qtot^2 + 0.004107 \times Qtot + 137.2$
Les résultats ne correspondent plus du tout au fichier Excel, nous avons des taux d'écarts entre les puissances de plus de 400% ! On peut aussi remarquer que les valeurs turbinées sont aberrantes. En effet, 3 turbines sont au maximum, la quatrième prends le reste et la dernière ne tourne pas.

```
Débit total : 549 Elevation amont : 172.11
Débit turbine 1 : 9 m3/s        134.412
Débit turbine 2 : 180 m3/s      145.471
Débit turbine 3 : 0 m3/s        134.277
Débit turbine 4 : 180 m3/s      0.000
Débit turbine 5 : 180 m3/s      127.055
Débit vanne : 0 0.000
Puissance produite : 781.9920351584356 MW       154.828607083747
Écart : 405.07%
```

FIGURE 4.7 – Résultat d'un calcul avec la fonction linéaire

Malheureusement, nous n'avons pas pu tester des fonctions de productions moins bien approximées que celle que nous avons, car la licence de matlab n'était plus valable lors des tests. Mais au vu de l'énorme changement visible seulement avec la fonction d'évaluation de l'élévation aval, nous ne pouvons qu'imaginer un changement radical et négatif si l'on détériore les fonctions de productions.

Chapitre 5

Discussion

5.1 Modularité de l'algorithme

Notre algorithme à été implémenté de manière à ce que presque tous les paramètres soient modulables. C'est-à-dire que la discrétisation du débit est une variable globale, ainsi on peut modifier sa valeur et l'algorithme se comportera toujours correctement. La valeur maximale de débit allouée par turbine est aussi une variable globale modifiable facilement. Les turbines qui ne doivent pas fonctionner pour une simulation sont dans un tableau binaire en variable globale (0 si la turbine ne fonctionne pas ou 1 si elle fonctionne). Ainsi, on peut lancer plein de simulations avec des paramètres différents pour essayer d'avoir un modèle qui correspond le mieux possible a la réalité.

5.2 Retour sur les résultats

Pour les résultats, nous pensons qu'ils sont très bons, car lors des tests que nous avons réalisés, on se retrouve à peu près dans la même fourchette de valeur que sur le fichier de Rio Tinto. Dans les moments ou les 5 turbines fonctionnent dans le fichier, sur notre algorithme les 5 turbines fonctionnent aussi. Si une turbine n'est pas en fonctionnement dans le fichier, dans notre algorithme une turbine, ne tourne pas non plus. De plus, lorsque l'on doit ouvrir la vanne de

déversement, notre algorithme propose aussi d'ouvrir la vanne de déversement aux mêmes moments que dans le fichier fourni par Rio Tinto.

5.3 Hypothèses requises pour terminer le travail

Bien que notre modèle d'optimisation fournisse des résultats très satisfaisants, nous avons remarqué que les turbines allumées/éteintes ne sont pas les mêmes. C'est peut-être parce que l'algorithme d'optimisation qu'utilise Rio Tinto est prévu pour ne pas toujours allumer et éteindre les turbines en plus d'optimiser le débit ? En effet, si une turbine effectue des cycles d'arrêt et de marche, on risque de l'endommager et de se retrouver dans un cas d'usure prématurée, c'est pour cela que nous pensons que les algorithmes de Rio Tinto font attention à ce genre de détail. Pour ajouter cela à notre algorithme, il faudrait ajouter une pénalité lors de l'allumage d'une turbine qui est éteinte, et une pénalité à l'extinction de celle ci. En plus de cela, on pourrait ajouter des pénalité si le débit est trop hétérogène entre les turbines.

5.4 Difficultés rencontrées

La première difficulté que nous avons rencontrée, c'est le fait qu'un de nos membres n'avait que très peu touché à la programmation et pas du tout à python, il a donc fallu un grand temps d'adaptation à la technologie.

La seconde difficulté que nous avons rencontrée est quant à elle liée à la programmation dynamique. En effet, après avoir programmé notre première version de l'algorithme nous nous sommes rendu compte qu'après le backwardpass, la turbine numéro 1 utilisait le reste du débit et turbinait tout même si ce n'était pas optimal. À cette étape-là du projet, nous n'avions pas pensé à la vanne de déversement, après avoir cherché longuement une solution à notre problème, nous nous sommes rendus compte que rajouter une table de programmation dynamique pour cette vanne pourrait résoudre notre problème. Ainsi, nous avons ajouté la vanne à notre algorithme de programmation dynamique (une étape de plus). Cette vanne

n'est autre qu'une turbine avec une fonction de production telle que $P_{vanne}(Q) = 0$

Chapitre 6

Conclusion et travaux futurs

Pour conclure sur ce travail numéro 2, nous avons beaucoup progressé dans la compréhension de la programmation dynamique en réalisant cet algorithme, le projet est très complet au niveau programmation dynamique dans la mesure où l'on devait implémenter un nombre d'étapes variable (turbines éteintes) et une limite de débit par turbine qui fait que nous ne sommes pas obligés d'utiliser toutes les ressource pour trouver la solution optimale. Prochainement, nous allons réaliser une interface graphique à notre logiciel pour le rendre utilisable facilement par des personnes qui ne sont pas familières avec la console. Et grâce à cette interface graphique, les options de modularité proposées par notre algorithme pourront être exploitées au maximum.

Annexe A

Fonction de production des turbines

A.0.1 Turbine 1

$P = 0.08574 - 0.002519H - 0.1976Q + 0.008179HQ + 0.002893Q^2 + 7.395 \times 10^{-6}HQ^2 - 1.196 \times 10^{-5}Q^3$

Avec H la hauteur de chute et Q le débit unitaire de la turbine.

A.0.2 Turbine 2

$P = 0.8117 - 0.02372H - 0.2443Q + 0.006487HQ + 0.003843Q^2 + 2.211 \times 10^{-5}HQ^2 - 1.672 \times 10^{-5}Q^3$

Avec H la hauteur de chute et Q le débit unitaire de la turbine.

A.0.3 Turbine 3

$P = -0.025 + 0.0009633H - 0.2158Q + 0.006349HQ + 0.003545Q^2 + 2.217 \times 10^{-5}HQ^2 - 1.576 \times 10^{-5}Q^3$

Avec H la hauteur de chute et Q le débit unitaire de la turbine.

A.0.4 Turbine 4

$P = -0.04626 + 0.001767H - 0.1908Q + 0.004949HQ + 0.003545Q^2 + 3.488 \times 10^{-5}HQ^2 - 1.698 \times 10^{-5}Q^3$

Avec H la hauteur de chute et Q le débit unitaire de la turbine.

A.0.5 Turbine 5

$P = 0.293 - 0.008025H - 0.1833Q + 0.008081HQ + 0.002709Q^2 + 1.955 \times 10^{-5}HQ^2 - 1.325 \times 10^{-5}Q^3$

Avec H la hauteur de chute et Q le débit unitaire de la turbine.