Optimisation de la production d'une centrale hydroélectrique
Projet de session 1

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

Hebs02629605, Seuj18099601, Patl21039604

13 mars 2019

Maîtrise informatique (3037)

Cours 8INF912 - Résolution de problèmes industriels

Table des matières

Table des figures

Introduction

Il y a quelque temps, l'entreprise X, qui est bien établie au Saguenay, a fait un appel d'offres pour octroyer un contrat très intéressant sur le développement de logiciels pour la gestion de la production hydroélectrique de leurs centrales. En tant qu'entreprise de consultation en informatique, nous sommes intéressé par cette demande, malgré le manque d'expérience en gestion de production hydroélectrique, car ce type d'énergie est très intéressant.

L'hydroélectricité est la source majeure d'énergie au Québec. En effet, 12% du territoire est recouvert de lacs et de rivières donc 97% de l'énergie produite provient de l'hydroélectricité. Cette énergie est d'autant plus appréciée parce que c'est une énergie propre est renouvelable [1].

Hydro-Québec, le principal producteur d'hydroélectricité de la région possède 62 centrales hydroélectriques qui possèdent une ou plusieurs turbines et sont de deux types [1]. En effet, ces centrales peuvent être au fil de l'eau, alors la production d'énergie est principalement influencée par le débit de la rivière, ou à réservoir, où la production d'énergie est contrôlée par la réserve d'eau présente dans le réservoir de la centrale. Le principe de fonctionnement est similaire pour les deux types de centrales hydroélectriques. En effet, les pales de la turbine sont poussées par la force de l'eau qui la fait donc tourner. Cela produit alors de l'énergie mécanique que l'on transforme en énergie électrique grâce à un alternateur, également présent dans les centrales. Voici un schéma d'une centrale hydroélectrique :

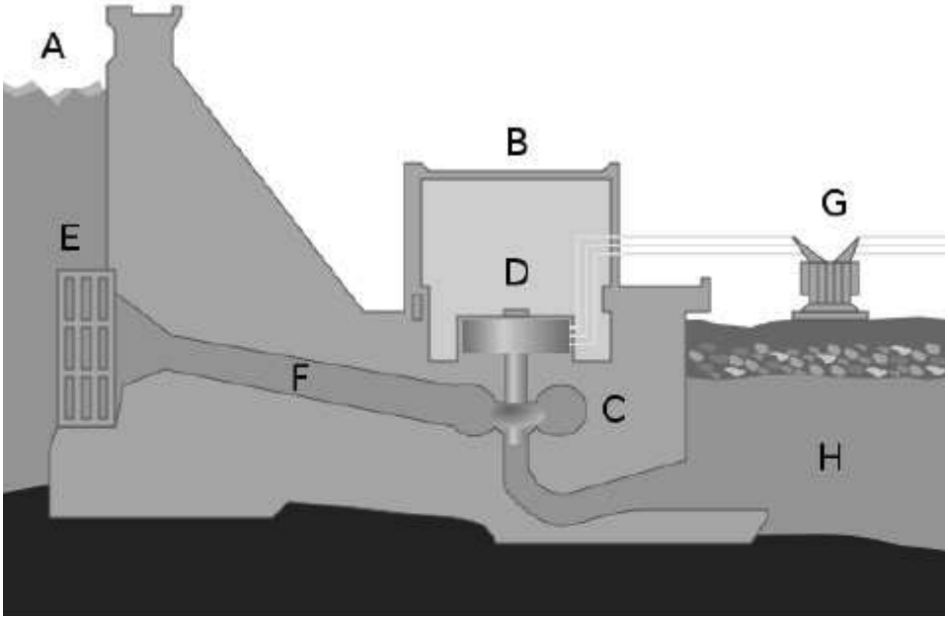


Figure 1 – Vue en coupe d'une centrale hydroélectrique

Légende :

- A - Brief amont : hauteur d'eau maximale du barrage
- B - Centrale hydroélectrique
- C et D - Groupe turbo-alternateur (turbine et alternateur)
- E - Grille
- F - Conduit permettant d'amener l'eau retenue par le barrage à la turbine
- G - Lignes de transport
- H - Brief aval : hauteur d'eau maximale à la sortie

Pour répondre à l'appel d'offres, nous nous intéressons à la centrale Chute-Savane qui comptabilise 5 turbines et fait partie du système du Saguenay Lac-St-Jean (SLSJ). L'entreprise X nous a fourni les données réelles de production entre 2013 et 2017, aux 2 minutes, de cette centrale. À partir de ces données, nous modéliserons un problème pour optimiser la production d'hydroélectricité dans cette centrale. Pour cela, nous modéliserons les fonctions de production hydroélectrique et les fonctions de production avale de chacune des turbines. Nous modéliserons et implémenterons le problème d'optimisation à court terme, ainsi qu'un algorithme de programmation dynamique permettant le chargement optimal des groupes. Enfin, nous créerons une page web permettant d'afficher les résultats trouvés grâce aux outils précédemment implémentés.

Dans ce premier rapport, nous nous contenterons de modéliser l'élévation avale de la centrale en fonction du débit total de cette dernière, ainsi que les fonctions de productions de chacune des turbines.

1 Méthode de modélisation

Dans cette partie, les principales méthodes de modélisation de droites, à savoir l'interpolation polynomiale, la régression polynomiale et une autre méthode de modélisation mettant en jeu la régression linéaire seront présentées [5, 6]. On considère dans ce qui suit $n+1, n \in \mathbb{N}_+$ données avec pour toute donnée $i \in \mathbb{N}_+, i < n+1$ une abscisse $x_i \in \mathbb{R}$ et une ordonnée $y_i \in \mathbb{R}$.

1.1 Interpolation polynomiale

L'interpolation polynomiale est une méthode de modélisation de droite qui vise à produire une fonction polynomiale de telle sorte que cette dernière passe par tous les points de nos données. L'interpolation polynomiale profite en fait du théorème suivant :

Soit $n \in \mathbb{N}_+$, il existe un et un seul polynôme P_n de degré inférieur ou égal à n pour lequel :

$$P_n(x_i) = y_i, \forall i \in \{0, \dots, n\} \tag{1}$$

Il existe deux méthodes pour trouver les coefficients du polynôme interpolateur, l'interpolation de Newton et l'interpolation de Lagrange. En pratique, l'interpolation de Newton est celle qui est la plus utilisée car elle permet facilement d'évaluer le polynôme interpolateur en moins d'opérations à l'aide de l'algorithme d'Hörner. L'interpolation de Newton, quant à elle, s'appuie sur le fait que le polynôme interpolateur puisse s'écrire :

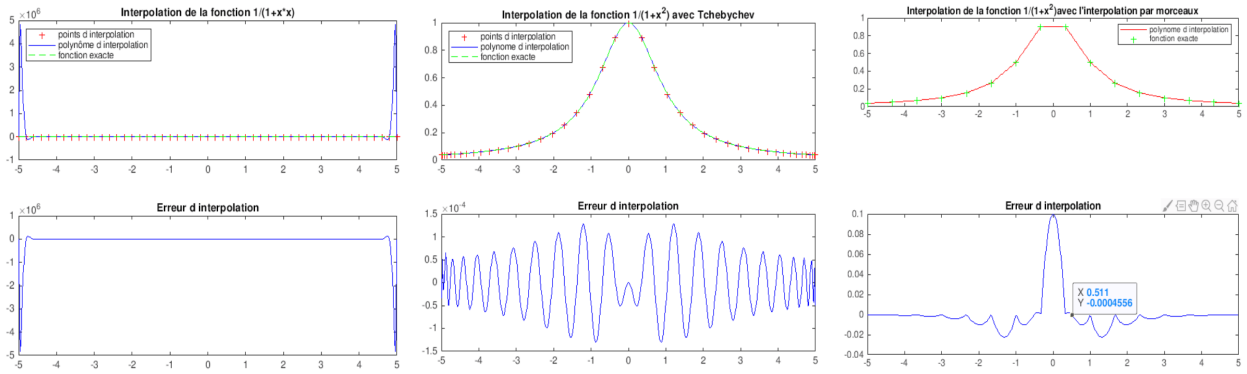
$$P_n(x) = \sum_{i=0}^{n} [y_0, y_1, \dots, y_i] \prod_{k=0}^{i-1} (x - x_k) \tag{2}$$

$$\text{Avec } [y_0, y_1, \dots, y_i] = \frac{[y_1, y_2, \dots, y_k] - [y_0, y_1, \dots, y_{k-1}]}{x_k - x_0} \text{ et } [y_i] = y_i, \forall i \in \{0, \dots, n\} \tag{3}$$

Les termes entre crochets sont appelés "différences divisées", elles sont définies récursivement mais peuvent être calculées de manière itérative.

Parmi les problèmes observés avec l'interpolation de Newton figurent le phénomène de Runge (voir 2a) et le comportement parfois très oscillatoire des polynômes de hauts degrés entre les points d'interpolation. Ce phénomène décrit le fait que l'approximation soit très mauvaise au bord de l'intervalle d'interpolation.

Face à ce problème, deux techniques sont principalement utilisées, les points de Chebyshev (voir 2b) et l'interpolation par morceaux (voir 2c). Très succinctement, la méthode des points de Chebyshev consiste en fait à placer plus de points d'interpolation près du bord de l'intervalle d'interpolation, qu'au centre. L'interpolation par morceaux consiste, elle, à interpoler les données sur plusieurs sous-intervalles de l'intervalle d'interpolation. On parle aussi de *spline interpolation*.



(a) Phénomène de Runge (b) Points de Chebyshev (c) Interpolation par morceaux

FIGURE 2 – Interpolation polynomiale

1.2 Régression polynomiale

La régression polynomiale est un méthode de modélisation qui essaye d'approximer le mieux possible la fonction à modéliser à l'aide d'un polynôme. Si ce polynôme est de degré 1, la régression est linéaire. Quand le degré du polynôme de la régression augmente, généralement, le polynôme se rapproche de celui obtenu avec l'interpolation. Dans la suite, on décrira comment le polynôme de la régression est trouvé. Le problème consiste alors à trouver le polynôme $P(x) = \sum_{i=0}^{m} \beta_i x^i,\ m < n,\ \beta_i \in \mathbb{R}\ \forall i \in \{0, \ldots, m\}$ qui minimise la somme du carré des $\epsilon_k \in \mathbb{R},\ k \in \{0, \ldots, n\}$ (méthode des moindres carrés) dans le système d'équations :

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^m \\ 1 & x_1 & x_1^2 & \ldots & x_1^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^m \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix} + \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix} \tag{4}$$

L'intérêt des moindres carrés par rapport à la somme des valeurs absolues par exemple, c'est qu'elle va plus fortement pénaliser les grands écarts, puisqu'ils seront mis au carré et ainsi garantir que toutes les données se situent bien relativement proche du polynôme de la régression. Il s'avère qu'il existe une solution analytique au problème d'optimisation énoncé ci-dessus dans le cas où l'on emploie la méthode des moindres carrés. Cette solution est la suivante :

$$\hat{\beta} = \left(X^T X\right)^{-1} X^T y \tag{5}$$

Il est aussi possible, tout comme pour l'interpolation, de diviser l'ensemble des données en plusieurs sous-ensembles et d'effectuer une régression polynomiale sur chacun de ces sous-ensembles. Ce type de régression est appelé *spline*. Les *splines* quadratiques imposent aux polynômes de deux sous-ensembles adjacents d'avoir les mêmes valeurs de fonction et de dérivée première sur leur bord commun. Les *splines* cubiques imposent en plus une même valeur pour la dérivée seconde des deux polynômes.

1.3 *Underfitting* et *overfitting*

Le degré du polynôme de la régression polynomiale est un paramètre très important. S'il est trop grand, le polynôme de la régression va avoir tendance à trop coller aux données, c'est de l'*overfitting*. Au contraire, s'il est trop faible, le polynôme de la régression ne va pas assez décrire nos données,

c'est de l'*underfitting*. Pour illustrer ces concepts, on a généré des données aléatoirement autour d'une fonction logarithmique et on a ensuite procédé, comme on peut le voir sur la figure 3, à une régression polynomiale de degré 1,2 et 5. Il est clair que la régression linéaire est en *underfitting* tandis que la régression polynomiale de degré 5 est en *overfitting* puisque les données sont en fait trop décrites, le comportement de la fonction logarithmique est perdu. Le degré le plus approprié ici semble être le degré 2.

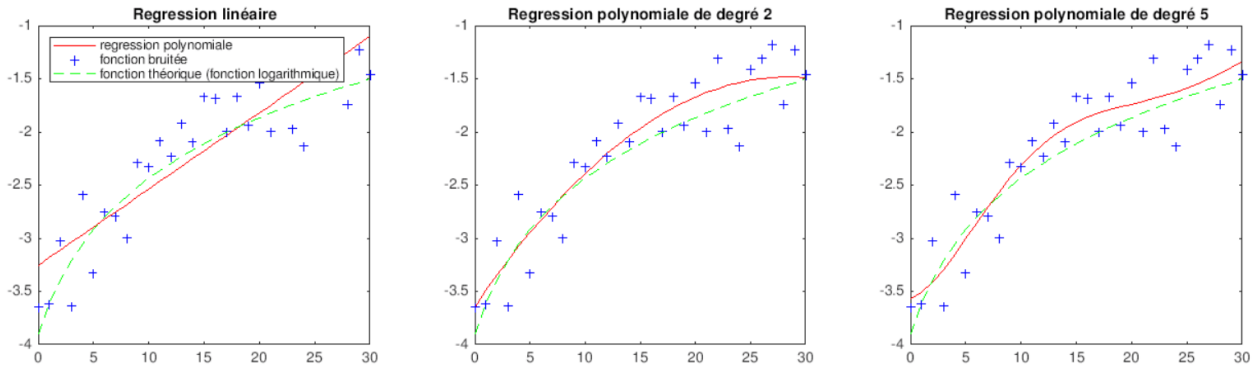


FIGURE 3 – Méthode de linéarisation délinéarisation

1.4 Autre technique de modélisation non linéaire

Une autre méthode de modélisation des données consiste à "linéariser" ces dernières, c'est-à-dire trouver une fonction f qui en étant appliquée aux données semble fournir une dépendance linéaire entre les y_i et les x_i. Il suffit, après cela, d'effectuer une régression linéaire puis à appliquer f^{-1} sur le polynôme obtenu avec la régression pour obtenir un modèle des données initiales.

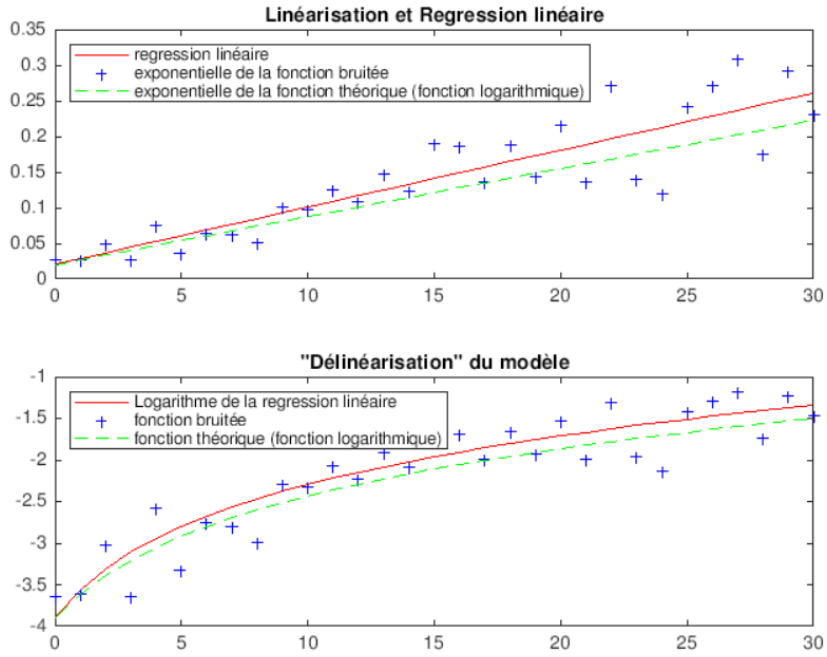


FIGURE 4 – Régression polynomiale de degré 1,2 et 5

Le résultat ainsi obtenu s'est avéré meilleur que celui obtenu avec la régression polynomiale uniquement. Il se rapproche en effet beaucoup plus de la courbe théorique logarithmique de laquelle dérivent les données.

2 Méthodologie et résultats

2.1 Chargement des données

La première étape de notre travail a été de charger les données mises à disposition dans MATLAB. Nous avons ainsi choisi de stocker les valeurs d'élévation, de débit total et de niveau d'eau en amont dans des vecteurs `elev`, `qtot` et `nivamont`. Puis nous avons stocké les différentes valeurs de $Q_1, ...Q_5$ et $P_1, ..., P_5$ dans deux matrices `Q` et `P` de dimensions 43824×5.

On peut ainsi par exemple accéder aux valeurs de P_i par la commande `P(:,i)`.

```
clear all;
close all;
load data_savane.mat;
data_savane = ChuteSavaneDonnees20132017;

%% Donnees:
elev = data_savane(:,2);
qtot = data_savane(:,3);
nivamont = data_savane(:,6);
Q = [data_savane(:,7), data_savane(:,9), data_savane(:,11),
data_savane(:,13), data_savane(:,15)];
P = [data_savane(:,8), data_savane(:,10), data_savane(:,12),
data_savane(:,14), data_savane(:,16)];
```

Les données sont ensuite formatée en Array :

```
%% Formatage des données en Array:
elev = table2array(elev);
qtot = table2array(qtot);
nivamont = table2array(nivamont);
Q = table2array(Q);
P = table2array(P);
```

Une fois les données chargées et formatées, nous pouvons passer à la première étape de notre travail de modélisation.

2.2 Modélisation de l'élévation avale en fonction du débit total

Nous avons dans un premier temps affiché le graphe des valeurs d'élévation avale en fonction du débit total.

```
figure;
plot(qtot,elev, "r+");
```

Nous pouvons clairement voir sur la figure 5 que les données sont concentrées et peuvent être approximées par une fonction polynomiale. Nous avons alors utilisé le module *Curve Fitting Tool* pour spécifier le polynôme en question.

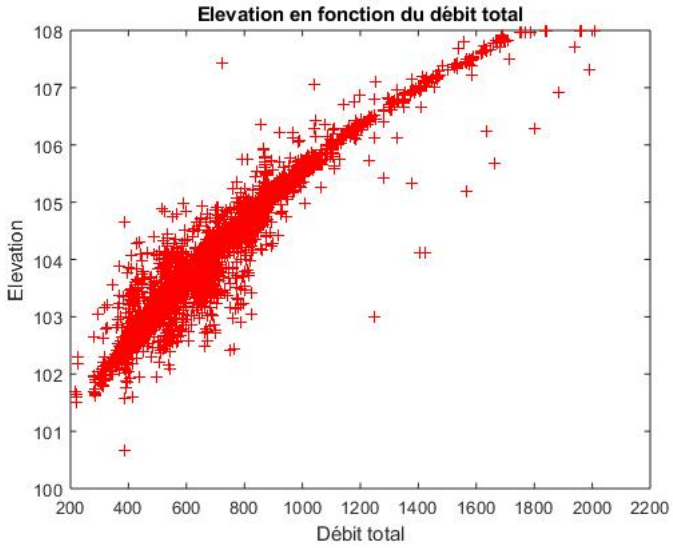


FIGURE 5 – Graphe de l'élévation avale en fonction du débit total

Après suppression des données aberrantes à la main, nous avons pris le choix d'approximer le nuage de points par un polynôme offrant un coefficient de détermination au moins égal à 0.97 de façon à disposer d'un modèle satisfaisant. Nous avons alors choisi un polynôme de degré 2. Le polynôme de degré 1 ne permettait pas d'atteindre la contrainte précédente, tandis qu'un polynôme de degré 3 offrait trop peu d'amélioration par rapport à celui de degré 2.

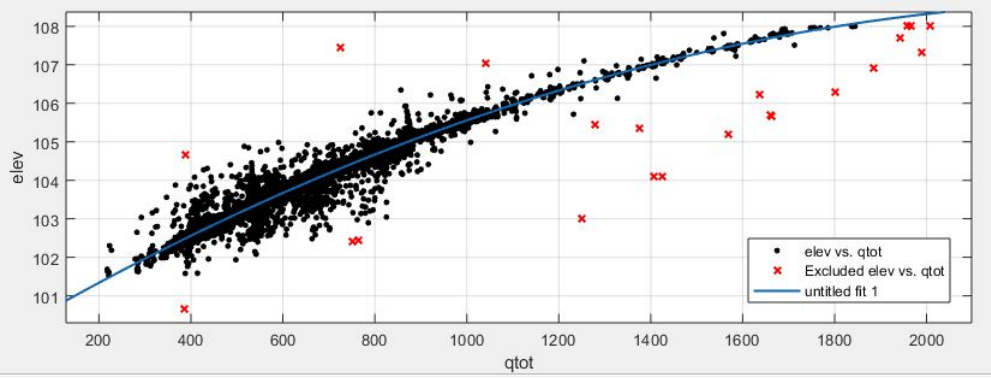


FIGURE 6 – Curvefitting de l'élévation avale en fonction du débit total

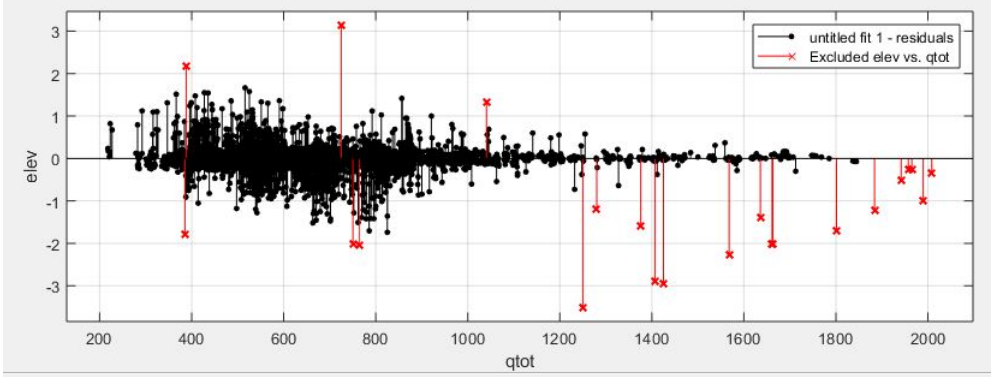


FIGURE 7 – Résidus des données d'élévation avale

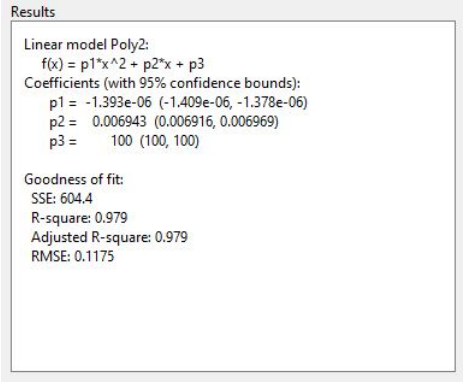


FIGURE 8 – Résultats de l'approximation polynomiale de l'élévation avale

Nous pouvons voir sur le graphique des résidus que les données sont concentrées autour de 0, et sur la toolbox que le coefficient de détermination est de 0.979, ce qui valide notre modèle.

La fonction finale est donc :

$$f_{elev}(q_{tot}) = a_0 + a_1 q_{tot} + a_2 q_{tot}^2 \tag{6}$$

avec

$$\begin{aligned} a_0 &= -1.393 \times 10^{-6} \\ a_1 &= 0.006943 \\ a_2 &= 100 \end{aligned}$$

2.3 Modélisation de la puissance en fonction du débit turbiné et de la hauteur de chute nette

Nous avons ensuite cherché a modéliser la puissance de chaque unité en fonction du débit turbiné et de la hauteur de chute nette associé. La première étape a été de calculer pour chaque turbine, la hauteur de chute nette selon la formule [1] :

$$h_{nette} = e_{am} - e_{av}(Q_{tot}) - 0,5.10^{-5} Q_i^2 \tag{7}$$

Nous avons ainsi créé une matrice `H` de taille 43824×5 dans laquelle nous calculons et stockons chaque valeur de hauteur nette pour chacune des turbines en fonction du débit correspondant.

```
H = nivamont - elev - 0.5e-5*Q.*Q;
```

Nous nous sommes ensuite attelés à la mise en équation des formules de puissances pour chaque turbines, à l'aide du toolbox *Curve fitting tool*. Nous avons dans un premier temps essayé de supprimer les quelques valeurs abbérantes sur chacun des modèles mais nous nous sommes apperçu qu'elles n'influencait presque pas le résultat de l'approximation polynomiale. Nous avons donc fait le choix de ne pas traiter ces valeurs abbérantes.

Pour chaque modèle, nous avons donc entré dans le toolbox, les vecteurs de hauteur de chute nette, de débit turbiné et de puissance pour les axes respectifs x, y et z. Puis, nous avons fait varier les puissance du polynôme approximateur d'arité 2 avec le critère suivant : avoir un coefficient de détermination R^2 d'au moins 0.999 tout en minimisant les degrés du polynôme.

Nous avons ainsi déterminer un polynôme

- P_1 de degré 1 en h et 2 en q
- P_2 de degré 1 en h et 3 en q
- P_3 de degré 1 en h et 1 en q
- P_4 de degré 1 en h et 3 en q
- P_5 de degré 1 en h et 2 en q

pour les puissance respective P_1, P_2, P_3, P_4 et P_5.

Voici deux exemples de fitting pour P_1 et P_2 :

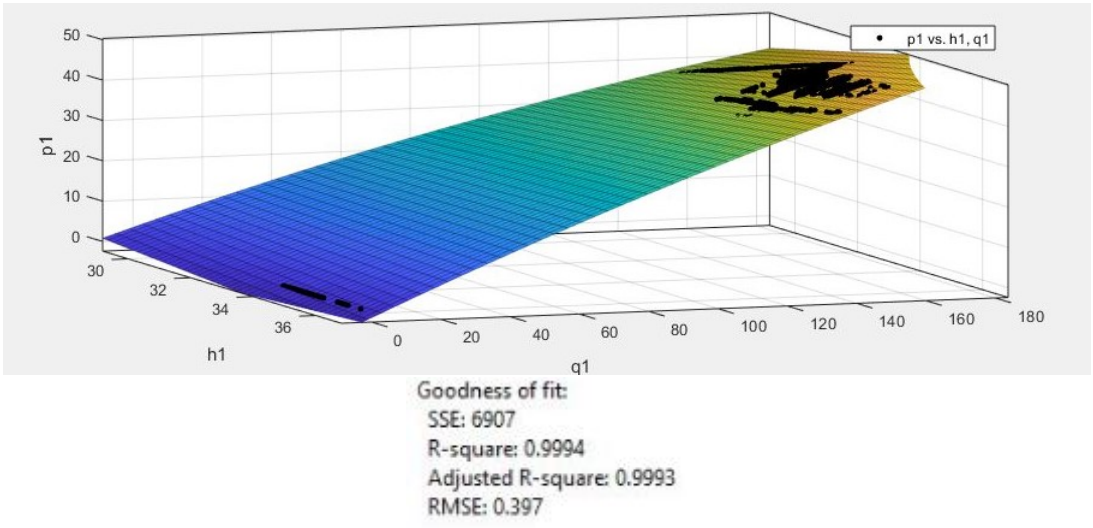


FIGURE 9 – Graphe de P1 en fonction de Q1 et H1

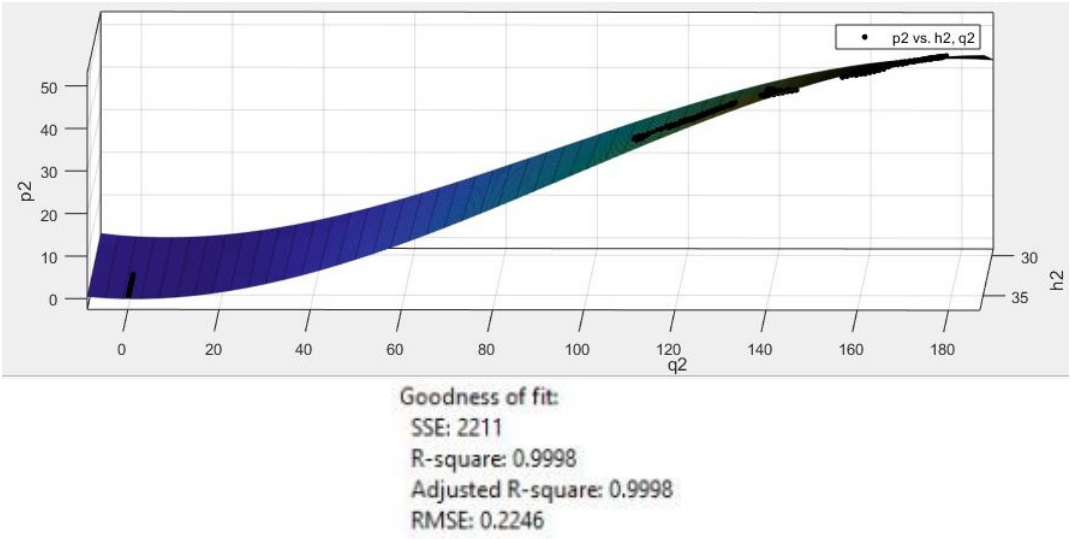


FIGURE 10 – Graphe de P2 en fonction de Q2 et H2

Les autres graphes sont disponibles en annexes.

Voici finalement les polynômes obtenus pour les expressions des puissances :

$$P_1(h,q) = 20,44 - 0,5908h - 0,138q + 0,01402hq - 0,0003412q^2 \quad (8)$$

$$P_2(h,q) = 0,6415 - 0,01847h - 0,1848q + 0,004656hq + 0,003525q^2 + 3,52.10^{-5}hq^2 - 1,655.10^{-5}q^3 \quad (9)$$

$$P_3(h,q) = 4,33.10^{-13} - 7,289.10^{-14}h + 0,97q \quad (10)$$

$$P_4(h,q) = 17,91 - 0,512h - 0,4176q + 0,0106hq + 0,004513q^2 + 2,047.10^{-5}hq^2 - 1,827.10^{-5}q^3 \quad (11)$$

$$P_5(h,q) = 16,02 - 0,4612h - 0,08307q + 0,01352hq - 0,0004633q^2 \quad (12)$$

Il est a précisé que nous ne sommes pas parvenu à obtenir un coefficient égal à 0.999 pour le polynôme P_4, nous avons donc essayé d'approcher au maximum notre critère tout en minimisant le degré, nous sommes parvenu finalement à obtenir un coefficient de détermination de 0.9984 pour ce polynôme.

2.4 Implémentation des fonctions en matlab

Après avoir déterminé les polynômes, nous les avons implémentés sous forme de fonction matlab pour pouvoir les réutiliser ultérieurement.

Nous avons ainsi créer les fonctions `f_elev`, `f_p1`, `f_p2`, `f_p3`, `f_p4`, `f_p5` :

```
function [ elev ] = f_elev(qtot)
    p1 = -1.393e-06;
    p2 = 0.006943;
    p3 = 100;
    elev = p1*qtot.*qtot+p2*qtot+p3;
end

function [puissance] = f_p1(q, h)
    p = [20.44,
    -0.5908,
    -0.138,
    0.01402,
    -0.0003412];
    puissance = p(1) + p(2).*h + p(3).*q + p(4).*h.*q + p(5).*q.*q;
end

function [puissance] = f_p2(q, h)
    p = [0.6415,
    -0.01847,
    -0.1848,
    0.004656,
    0.003525,
    3.52e-05,
    -1.655e-05];
    puissance = p(1) + p(2).*h + p(3).*q + p(4).*h.*q +
        p(5).*q.*q +p(6)*h.*q.*q+p(7).*q.*q.*q;
end
```

```
function [puissance] = f_p3(q, h)
    p = [4.334e-13, -7.289e-14, 0.97];
    puissance = p(1) + p(2).*h + p(3).*q;
end

function [puissance] = f_p4(q, h)
    p = [17.91,
    -0.512,
    -0.4176,
    0.0106,
    0.004513,
    2.047e-05,
    -1.827e-05];
    puissance = p(1) + p(2).*h + p(3).*q + p(4).*h.*q +
        p(5).*q.*q +p(6).*h.*q.*q+p(7).*q.*q.*q;
end

function [puissance] = f_p5(q, h)
    p = [16.02,
    -0.4612,
    -0.08307,
    0.01352,
    -0.0004633];
    puissance = p(1) + p(2).*h + p(3).*q + p(4).*h.*q + p(5).*q.*q;
end
```

2.5 Validation du modèle

Nous avons ensuite cherché à confirmer nos choix précédent pour valider le modèle final. Intéressons nous d'abord à la fonction, qui modélise l'élévation avale en fonction du débit total. Nous avons tracé cette fonction à partir de la fonction `f_elev` que nous avons implémenté précédemment et l'avons superposé au nuage de points.

```
    figure;
    plot(qtot,elev, "r+");
    hold on;
    x = linspace(200,2000,1000);
    p = plot(x, f_elev(x));
    p.LineWidth = 2;
    title("Elevation en fonction du débit total");
    xlabel("Débit total");
    ylabel("Elevation");
```

Nous avons ensuite cherché à confirmer le modèle pour les 5 fonctions de puissance. Cela nous permet entre autres de confirmer qu'il n'y a pas eu d'erreur lors de la saisie des coefficients donnés par le curve fitting dans les différentes fonctions matlab de puissance. Pour ce faire, nous avons stocké dans une matrice 43824×5 les valeurs données par les fonctions matlab f_{pi} en fonction des valeurs des vecteurs H_i et Q_i correspondant. Puis nous avons calculé pour chaque $i \in \{1, 2, 3, 4, 5\}$,

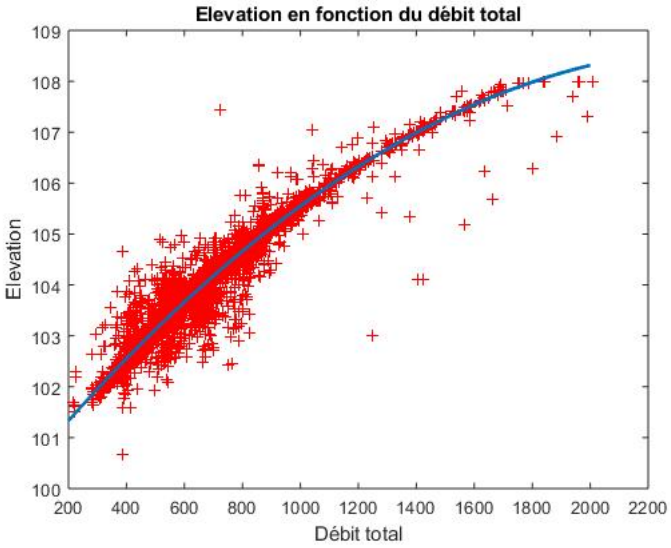


FIGURE 11 – Superposition du modèle f_elev et des données

le coefficient de corrélation entre les valeurs réelles du vecteur P_i et les valeurs données par les fonctions f_{pi}.

On rappelle que le coefficient de corrélation entre deux variables aléatoire X et Y est donné par la formule [3] :

$$r(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{13}$$

```
P_model(:,1) = f_p1(Q(:,1),H(:,1));
P_model(:,2) = f_p2(Q(:,2),H(:,2));
P_model(:,3) = f_p3(Q(:,3),H(:,3));
P_model(:,4) = f_p4(Q(:,4),H(:,4));
P_model(:,5) = f_p5(Q(:,5),H(:,5));

corr(P,P_model)
```

La commande `corr(P,P_model)` renvoie la matrice de corrélation de `P` et `P_model` donnée par la formule [4] :

$$C = \begin{pmatrix} r(P_1,P_{1m}) & r(P_1,P_{2m}) & r(P_1,P_{3m}) & r(P_1,P_{4m}) & r(P_1,P_{5m}) \\ r(P_2,P_{1m}) & r(P_2,P_{2m}) & r(P_2,P_{3m}) & r(P_2,P_{4m}) & r(P_2,P_{5m}) \\ r(P_3,P_{1m}) & r(P_3,P_{2m}) & r(P_3,P_{3m}) & r(P_3,P_{4m}) & r(P_3,P_{5m}) \\ r(P_4,P_{1m}) & r(P_4,P_{2m}) & r(P_4,P_{3m}) & r(P_4,P_{4m}) & r(P_4,P_{5m}) \\ r(P_5,P_{1m}) & r(P_5,P_{2m}) & r(P_5,P_{3m}) & r(P_5,P_{4m}) & r(P_5,P_{5m}) \end{pmatrix} \tag{14}$$

Les coefficients qui nous intéressent sont donc les coefficients de la diagonale de la matrice précédente.

Matlab renvoie le résultat suivant :

```
    0.9996   -0.1072    0.0034   -0.0486   -0.0986
   -0.1099    0.9999    0.0303    0.1208    0.1714
    0.0071    0.0292    1.0000    0.1379   -0.1282
   -0.0497    0.1161    0.1357    0.9985    0.2055
   -0.1027    0.1704   -0.1288    0.2037    0.9998
```

Ainsi :

$$r(P_1, P_{1,modèle}) = 0.9996$$
$$r(P_2, P_{2,modèle}) = 0.9999$$
$$r(P_3, P_{3,modèle}) = 1.0000$$
$$r(P_4, P_{4,modèle}) = 0.9985$$
$$r(P_5, P_{5,modèle}) = 0.9998$$

3 Discussion

L'approximation de l'élévation avale en fonction du débit total fut simple à effectuer. En effet, nous obtenons des résultats d'approximation très satisfaisants puisque nous avons $R^2 = 0{,}9763$. Lorsque nous visualisons la fonction d'approximation obtenue avec le nuage de points des données, cela semble très satisfaisant.

De même, nous obtenons des approximations très proches des données réelles pour l'ensemble des 5 fonctions de puissances associées aux 5 turbines. L'ensemble des R^2 sont tous aux alentours de 0,999.

Nous avons ainsi obtenu de très bonnes approximations pour l'ensemble de nos fonctions, même si nous avons fait le choix de ne pas supprimer de données aberrantes. En effet, nous avons essayé et, après comparaison, les données aberrantes étaient trop peu nombreuses pour avoir un réel impact sur la modélisation des fonctions. Nous n'avons pas spécialement eu de difficultés rencontrées. En effet, la première question que nous avons soulevée était à propos du filtrage des données. Nous nous demandions s'il était réellement utile de filtrer les données aberrantes, sachant qu'après visualisation des données, nous n'en relevions qu'une poignée noyée au milieu des dizaines de milliers restantes. Ainsi, nous avons décidé de faire un test sur la fonction pour laquelle nous avions une approximation la moins juste, c'est-à-dire la puissance associée à la turbine 4. Après comparaison, nous nous sommes rendu compte que l'impact des données aberrantes était minime. Ainsi, nous avons pris la décision de ne pas filtrer les données.

Enfin, pour déterminer les différents degrés d'approximation des fonctions de puissance, il nous a fallu choisir un critère à respecter qui nous permettrait de les fixer. Nous avons alors fait le choix que la condition à remplir pour choisir les degrés soit un $R^2 \geq 0.999$, tout en choisissant les degrés les plus petits possibles pour les deux paramètres. Nous avons alors rencontré une difficulté pour la fonction de puissance associée à la turbine 4. En effet, même en choisissant les degrés les plus hauts pour l'approximation polynomiale, nous n'obtenions pas un $R^2 \geq 0.999$. Ainsi, nous avons revu notre condition à la baisse pour ce polynôme uniquement en fixant $R^2 \geq 0.998$. Nous avons ainsi obtenu une approximation satisfaisante.

Conclusion

Nous avons donc réussi à modéliser de façon satisfaisante les fonctions de productions des 5 turbines ainsi que l'élévation avale en fonction du débit total à l'aide de la régression polynomiale.

Ces modèles de fonctions ainsi obtenus nous seront utiles dans des travaux futurs afin de déterminer les volumes des réservoirs, les turbines en fonctionnement ainsi que les débits à turbiner à chaque période pour maximiser la puissance globale produite par les centrales hydroélectriques. L'objectif final serait de fournir un outil de gestion des centrales étudiées dont l'utilisation serait facilitée par une interface, qui permettrait soit de maximiser la production totale d'énergie lorsque le débit total initial est imposé, soit de déterminer la répartition de ce débit afin d'atteindre la production totale recherchée.

Références

[1] S, Séguin. "Résolution de problèmes industriels - Cours 2 : Optimisation de la production hydroélectrique", présenté à l'Université du Québec à Chicoutimi dans le cours Résolution de Problèmes Industriels 8INF912, 2019.

[2] S, Séguin. "Résolution de problèmes industriels - Cours 5 : Méthodes numériques", présenté à l'Université du Québec à Chicoutimi dans le cours Résolution de Problèmes Industriels 8INF912, 2019.

[3] Contributeurs de Wikipédia. (2019, 12 mars). *Corrélation (statistiques) - Wikipédia, l'encyclopédie libre* [Internet]. Disponible à : `http://fr.wikipedia.org/w/index.php?title=Corr%C3%A9lation_(statistiques)`

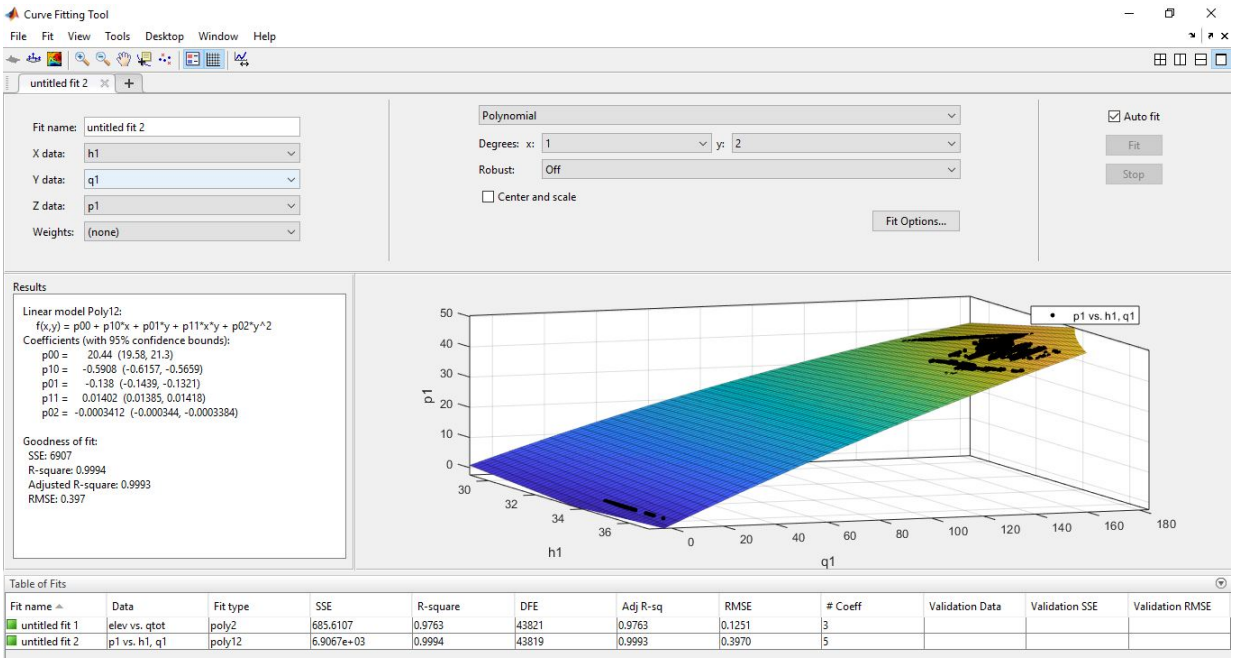
[4] The MathWorks, Inc. (2019, 12 mars). *Documentation : corrcoef - Correlation coefficients* [Internet]. Disponible à : `https://fr.mathworks.com/help/matlab/ref/corrcoef.html`

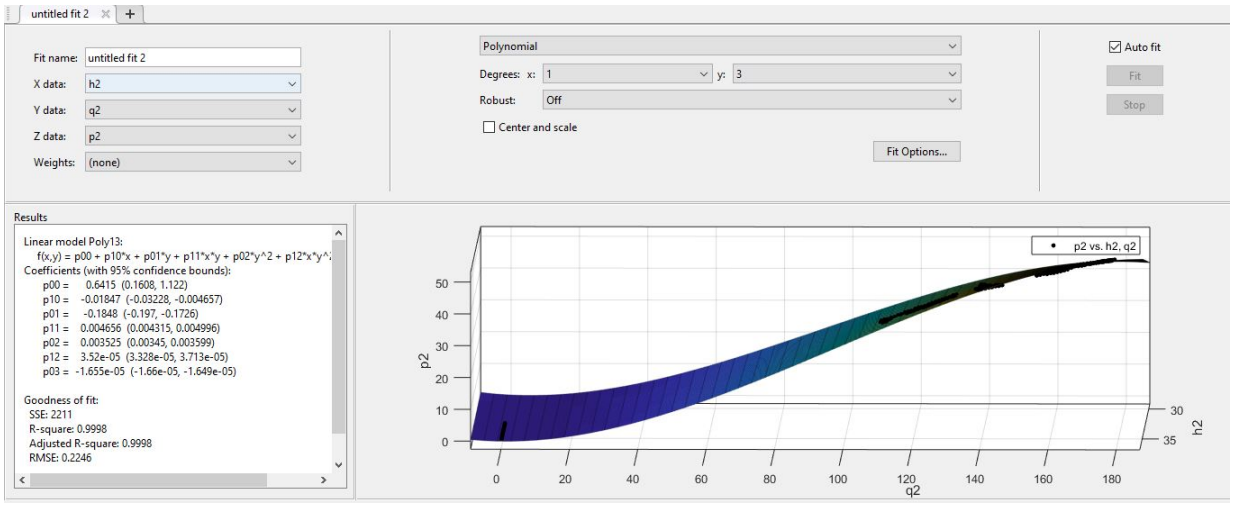
[5] L, Monsaingeon. (2017). *Analyse Numérique - Chapitre 4 :Interpolation et approximation polynomiale.* [Internet]. Disponible à : `http://www.iecl.univ-lorraine.fr/~Leonard.Monsaingeon/fichiers/enseignements/2016-2017/ENSEM_ANUM/Cours-04-Interpolation.pdf`

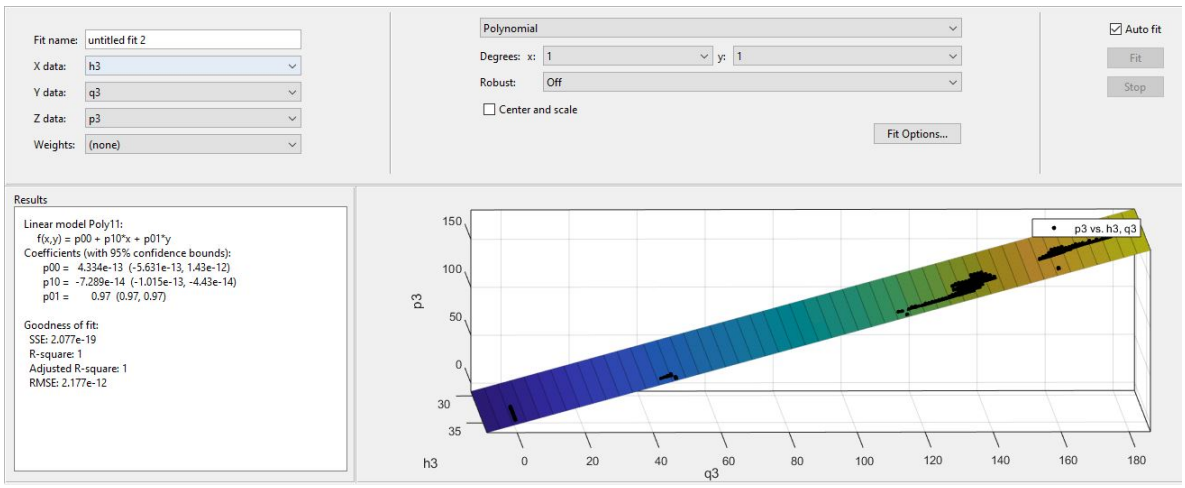
[6] Contributeurs de Wikipédia. (2019, 16 février) *Polynomial regression - Wikipédia, l'encyclopédie libre.* [Internet]. Disponible à : `https://en.wikipedia.org/wiki/Polynomial_regression`

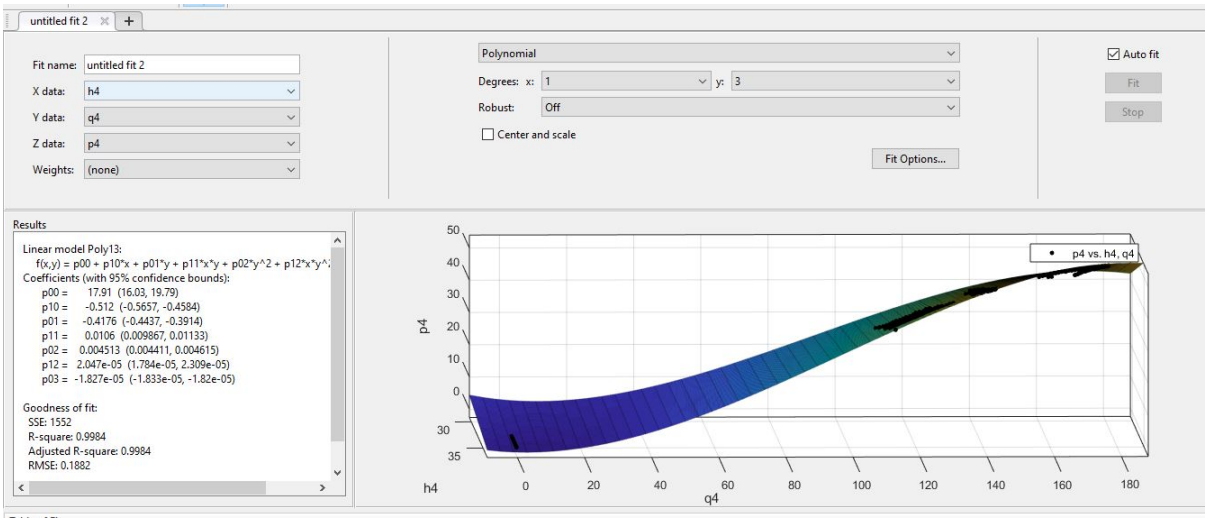
Annexes

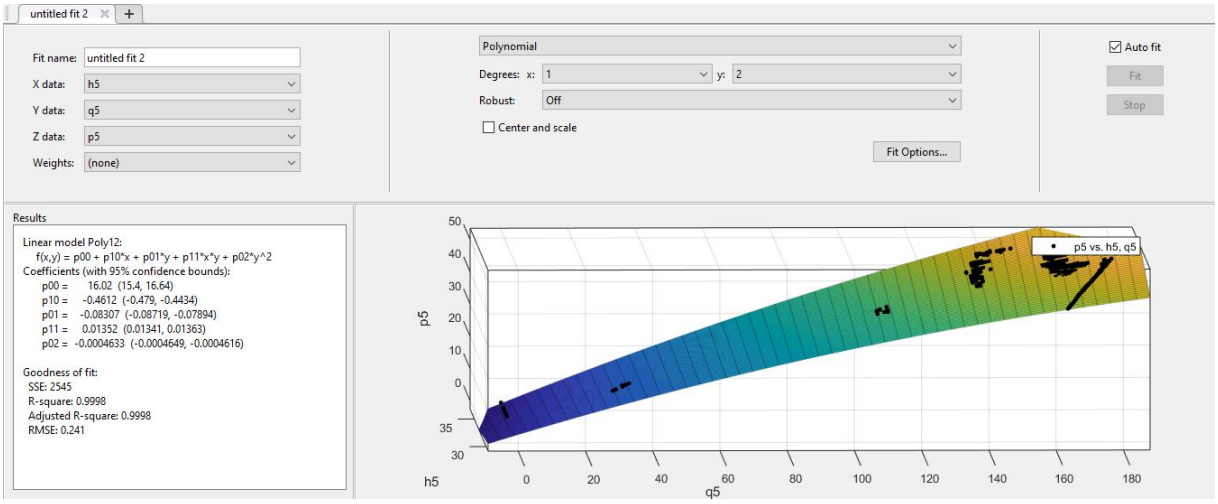
Visualisation de l'interface de Curvefitting pour les 5 fonctions puissance








untitled fit 2
Fit name: untitled fit 2
X data: h4
Y data: q4
Z data: p4
Weights: (none)
Polynomial
Degrees: x: 1 y: 3
Robust: Off
Center and scale
Fit Options...
Auto fit
Fit
Stop
Results
Linear model Poly13:
f(x,y) = p00 + p10*x + p01*y + p11*x*y + p02*y^2 + p12*x*y^
Coefficients (with 95% confidence bounds):
p00 = 17.91 (16.03, 19.79)
p10 = -0.512 (-0.5657, -0.4584)
p01 = -0.4176 (-0.4437, -0.3914)
p11 = 0.0106 (0.009867, 0.01133)
p02 = 0.004513 (0.004411, 0.004615)
p12 = 2.047e-05 (1.784e-05, 2.309e-05)
p03 = -1.827e-05 (-1.833e-05, -1.82e-05)
Goodness of fit:
SSE: 1552
R-square: 0.9984
Adjusted R-square: 0.9984
RMSE: 0.1882
p4 vs. h4, q4
p4
h4
q4


untitled fit 2
Fit name: untitled fit 2
X data: h5
Y data: q5
Z data: p5
Weights: (none)
Polynomial
Degrees: x: 1 y: 2
Robust: Off
Center and scale
Fit Options...
Auto fit
Fit
Stop
Results
Linear model Poly12:
f(x,y) = p00 + p10*x + p01*y + p11*x*y + p02*y^2
Coefficients (with 95% confidence bounds):
p00 = 16.02 (15.4, 16.64)
p10 = -0.4612 (-0.479, -0.4434)
p01 = -0.08307 (-0.08719, -0.07894)
p11 = 0.01352 (0.01341, 0.01363)
p02 = -0.0004633 (-0.0004649, -0.0004616)
Goodness of fit:
SSE: 2545
R-square: 0.9998
Adjusted R-square: 0.9998
RMSE: 0.241
p5 vs. h5, q5
p5
h5
q5